Bases statistiques et tests d'hypothèses avec R

Christophe Ambroise et Julien Chiquet

Université d'Évry val d'Essonne

7-9 avril 2014

`http://stat.genopole.cnrs.fr/~jchiquet/fr/initiation_R`

Intervenant

Equipe « Statistique & Génome »

`http://stat.genopole.cnrs.fr`

Christophe Ambroise

Statistique

`prenom.nom@genopole.cnrs.fr`

Première partie I

Introduction à la statistique

Qu'est ce que la statistique ?

Statistique

Activité qui consiste dans le recueil, le traitement et l'interprétation de données d'observation.

Population

- population, ensemble d'entités objet de l'investigation statistique, et non telle ou telle entité particulière.
- individus, définis comme les éléments d'une certaine population.

Différentes notion de population

- Dans certain cas la population de référence est finie et ses éléments peuvent être explicitement dénombrés
- la notion de population revêt parfois une signification plus abstraite (exemple population de malades)
- parfois la notion de population s'identifie avec celle de procédure de génération de données (données d'expression)

Variables

Chaque individu est décrit par un ensemble de variables :

- qualitative (sexe, nationalité, état matrimonial, ...) : les valeurs prises par le caractère sont les modalités
  - ordinale (notion d'ordre) : modalités intrinsèquement ordonnées
  - nominale : pas de structure d'ordre : par exemple le sexe.
- quantitative (taille, poids ...)
  - discrète
  - continue

Modes d'étude d'une population

Étude exhaustive

Dite par recensement.
L'étude d'une population de grande taille est souvent difficile voire impossible

Échantillon

Le processus de sélection d'un échantillon est l'échantillonnage Seule solution d'une le cas d'une population infinie

Inférence statistique

Processus visant à

- déduire de conclusions générales relative à la population totale
- à partir de la connaissance particlle relative à un nombre de cas particulier

Objectifs d'une étude statistique

1. synthétiser, résumer, structurer l'information :
   Statistique Descriptive ou Exploratoire
2. formuler ou valider des hypothèses relatives à la population totale :
   Statistique inférentielle

Le tableau de données

- n individus mesurés par p variables
- Tableau $X = (x_i^j) = \begin{pmatrix} x_1^1 & x_1^j & x_1^p \\ & \cdot & \cdot \\ x_i^1 & x_i^j & x_i^p \\ & \cdot & \cdot \\ x_n^1 & x_n^j & x_n^p \end{pmatrix}$
- Chaque variable est représentée par le vecteur $\boldsymbol{x}^j = (x_1^j, \ldots, x_n^j)'$
- Chaque individu est représenté par le vecteur $\boldsymbol{x}_i = (x_i^1, \ldots, , x_i^p)'$
- X : réalisation d'un échantillon de taille n du vecteur aléatoire de dimension p

$$\mathbf{X} = (X^1, \ldots, X^p)'$$

Deuxième partie II

Introduction à R

Qu'est-ce que R ?

En deux mots,

R est un logiciel de développement scientifique spécialisé dans le calcul et l'analyse statistique.

R est aussi

- un langage,
- un environnement,
- un projet open source (projet GNU),
- un logiciel multi-plateforme (Linux, Mac, Windows),
- la 18e lettre de l'alphabet.

Qu'est-ce que R ?

En deux mots,

R est un logiciel de développement scientifique spécialisé dans le calcul et l'analyse statistique.

R est aussi

- un langage,
- un environnement,
- un projet open source (projet GNU),
- un logiciel multi-plateforme (Linux, Mac, Windows),
- la 18e lettre de l'alphabet ☹.

Qu'est-ce que R ?

En deux mots,

R est un logiciel de développement scientifique spécialisé dans le calcul et l'analyse statistique.

R est aussi

- un langage,
- un environnement,
- un projet open source (projet GNU),
- un logiciel multi-plateforme (Linux, Mac, Windows),
- la 18e lettre de l'alphabet ☹.

Principales fonctionnalités

1. Gestionnaire de données
   - Lecture, manipulation, stockage.
2. Algèbre linéaire
   - Opérations classiques sur vecteurs, tableaux et matrices
3. Statistiques et analyse de données
   - Dispose d'un *grand* nombre de méthodes d'analyse de données (des plus anciennes et aux plus récentes)
4. Moteur de sorties graphiques
   - Sorties écran ou fichier
5. Système de modules
   - Alimenté par la communauté (+ de 2000 extensions !)
6. Interface facile avec `C/C++`, `Fortran`,...

Principales fonctionnalités

1. Gestionnaire de données
   - Lecture, manipulation, stockage.
2. Algèbre linéaire
   - Opérations classiques sur vecteurs, tableaux et matrices
3. Statistiques et analyse de données
   - Dispose d'un *grand* nombre de méthodes d'analyse de données (des plus anciennes et aux plus récentes)
4. Moteur de sorties graphiques
   - Sorties écran ou fichier
5. Système de modules
   - Alimenté par la communauté (+ de 2000 extensions !)
6. Interface facile avec `C/C++`, `Fortran`,...

Historique

Approche chronologique

1970s développement de S au Bell labs.

1980s développement de S-PLUS au AT&T. Lab

1993 développement de R sur le modèle de S par Robert Gentleman et Ross Ihaka au département de statistique de l'université d'Auckland.

1995 dépôts des codes sources sous licence GNU/GPL

1997 élargissement du groupe

2002 la fondation R dépose ses statuts sous la présidence de Gentleman et Ihaka

Développement entièrement bénévole

- « R development core team » (12aine de personnes)
- Participation de *nombreux* chercheurs (2000 packages)

Historique

Approche chronologique

1970s développement de S au Bell labs.

1980s développement de S-PLUS au AT&T. Lab

1993 développement de R sur le modèle de S par Robert Gentleman et Ross Ihaka au département de statistique de l'université d'Auckland.

1995 dépôts des codes sources sous licence GNU/GPL

1997 élargissement du groupe

2002 la fondation R dépose ses statuts sous la présidence de Gentleman et Ihaka

Développement entièrement bénévole

- « R development core team » (12aine de personnes)
- Participation de *nombreux* chercheurs (2000 packages)

Se tenir informé

1. La page web de la fondation R
   - les statuts, des liens, des références.
   - `http://www.r-project.org/`

2. La page web du CRAN (Comprehensive R Arxiv Network)
   - binaires d'installation, packages, documentations, ...
   - `http://cran.r-project.org/`

3. La conférence des utilisateurs de R :
   - annuelle, prochaine édition à Gaithersburg
   - `http://user2010.org/`

4. *The R journal* propose des articles sur
   - de nouvelles extensions, des applications, des actualités.
   - `http://journal.r-project.org/`

Se tenir informé

1. La page web de la fondation R
   - les statuts, des liens, des références.
   - `http://www.r-project.org/`

2. La page web du CRAN (Comprehensive R Arxiv Network)
   - binaires d'installation, packages, documentations, ...
   - `http://cran.r-project.org/`

3. La conférence des utilisateurs de R :
   - annuelle, prochaine édition à Gaithersburg
   - `http://user2010.org/`

4. *The R journal* propose des articles sur
   - de nouvelles extensions, des applications, des actualités.
   - `http://journal.r-project.org/`

Se tenir informé

1. La page web de la fondation R
   - les statuts, des liens, des références.
   - `http://www.r-project.org/`

2. La page web du CRAN (Comprehensive R Arxiv Network)
   - binaires d'installation, packages, documentations, ...
   - `http://cran.r-project.org/`

3. La conférence des utilisateurs de R : useR!
   - annuelle, prochaine édition à Gaithersburg
   - `http://user2010.org/`

4. *The R journal* propose des articles sur
   - de nouvelles extensions, des applications, des actualités.
   - `http://journal.r-project.org/`

Se tenir informé

1. La page web de la fondation R
   - les statuts, des liens, des références.
   - `http://www.r-project.org/`

2. La page web du CRAN (Comprehensive R Arxiv Network)
   - binaires d'installation, packages, documentations, ...
   - `http://cran.r-project.org/`

3. La conférence des utilisateurs de R : 
   - annuelle, prochaine édition à Gaithersburg
   - `http://user2010.org/`

4. *The R journal* propose des articles sur
   - de nouvelles extensions, des applications, des actualités.
   - `http://journal.r-project.org/`

Qualités et défauts de R

Plus ☺

1. Libre et gratuit,
2. Richesse des modules (en statistique),
3. Rapidité d'exécution,
4. Développement rapide (langage de scripts),
5. Syntaxe intuitive et compact,
6. Nombreuses possibilités graphiques.

Moins ☹

1. Aide intégrée succincte,
2. Debugger un peu sec,
3. Code parfois illisible (compacité),
4. Personnalisation des graphiques un peu lourde.

Qualités et défauts de R

Plus ☺

1. Libre et gratuit,
2. Richesse des modules (en statistique),
3. Rapidité d'exécution,
4. Développement rapide (langage de scripts),
5. Syntaxe intuitive et compact,
6. Nombreuses possibilités graphiques.

Moins ☹

1. Aide intégrée succincte,
2. Debugger un peu sec,
3. Code parfois illisible (compacité),
4. Personnalisation des graphiques un peu lourde.

Les concurrents plus ou moins directs

Les logiciels de développement scientifique sont spécialisés en

1. algèbre linéaire
   - `Matlab` (Mathworks), la référence,
   - `Scilab` (INRIA), l'alternative libre,
   - `Octave` (GNU), l'alternative open source ☺,
2. statistiques
   - `SAS` (SAS Inc.), la référence,
   - `S-PLUS` (TIBCO), le concurrent,
   - `R` (GNU), l'alternative open source ☺,
3. calcul symbolique
   - `Mathematica` (Wolfram), la référence,
   - `Maple` (Maplesoft), la référence aussi,
   - `Maxima` (GNU), l'alternative open source ☺.

Les concurrents plus ou moins directs

Les logiciels de développement scientifique sont spécialisés en

1. algèbre linéaire
   - `Matlab` (Mathworks), la référence,
   - `Scilab` (INRIA), l'alternative libre,
   - `Octave` (GNU), l'alternative open source ☺,
2. statistiques
   - `SAS` (SAS Inc.), la référence,
   - `S-PLUS` (TIBCO), le concurrent,
   - `R` (GNU), l'alternative open source ☺,
3. calcul symbolique
   - `Mathematica` (Wolfram), la référence,
   - `Maple` (Maplesoft), la référence aussi,
   - `Maxima` (GNU), l'alternative open source ☺.

Les concurrents plus ou moins directs

Les logiciels de développement scientifique sont spécialisés en

1. algèbre linéaire
   - `Matlab` (Mathworks), la référence,
   - `Scilab` (INRIA), l'alternative libre,
   - `Octave` (GNU), l'alternative open source ☺,
2. statistiques
   - `SAS` (SAS Inc.), la référence,
   - `S-PLUS` (TIBCO), le concurrent,
   - `R` (GNU), l'alternative open source ☺,
3. calcul symbolique
   - `Mathematica` (Wolfram), la référence,
   - `Maple` (Maplesoft), la référence aussi,
   - `Maxima` (GNU), l'alternative open source ☺.

Matlab versus R

- Obtenir de l'aide

```
help -i                    help.start ()
help                       help(help)
help sort                  help(sort) _or_ ?sort
```

- Séquence de vecteurs

```
1:10                       1:10  _or_ seq(10)
1:3:10                     seq(1,10,by=3)
10:-1:1                    10:1
linspace(1,10,7)           seq(1,10,length=7)
```

- Manipulation de vecteurs

```
a=[2 7 8 5]                a <- c(2,7,8,5)
a=a[3:4                    a <- a[c(3,4)]
adash=[2 3 4 5]'           adash <- t(c(2,3,4,5))
```

Installation

Rendez-vous sur la page du CRAN `http://cran.r-project.org/`

Macintosh

Télécharger `R-2.10.1.pkg`, cliquer.

Windows

Télécharger `R-2.10.1-win32.exe`, cliquer (prier).

Linux

Systèmes supportant `apt` (Debian, Ubuntu, ...)

```
$ sudo apt-get update
$ sudo apt-get install r-base
```

Lancer R

Dans un terminal, taper 'R'

Premiers pas

```
$ R
R version 2.10.1 (2009-12-14)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
[...]
Tapez 'demo()' pour des démonstrations, 'help()' pour l'aide
en ligne ou 'help.start()' pour obtenir l'aide au format HTML.
Tapez 'q()' pour quitter R.
> 1+1
[1] 2
```

Sortez moi de là !

```
> q()
Save workspace image? [y/n/c]:y
```

$\rightsquigarrow$ Sauve l'environnement et le réouvre la prochaine fois

Organiser un projet R
À calquer lors des travaux dirigés

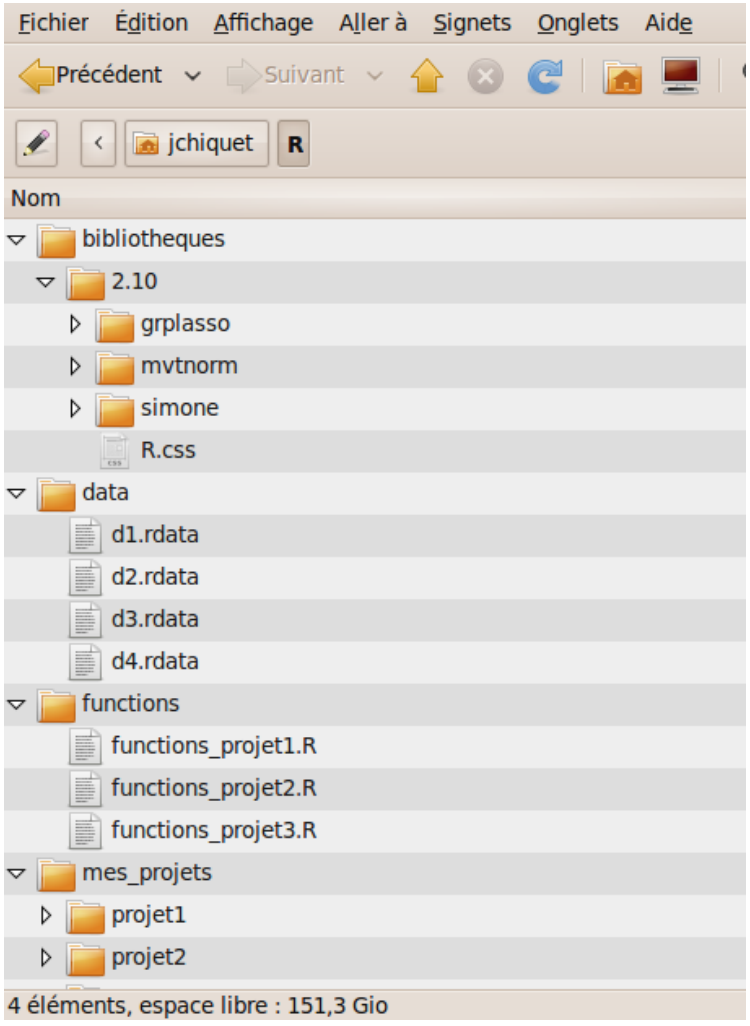


FIGURE : Arborescence type

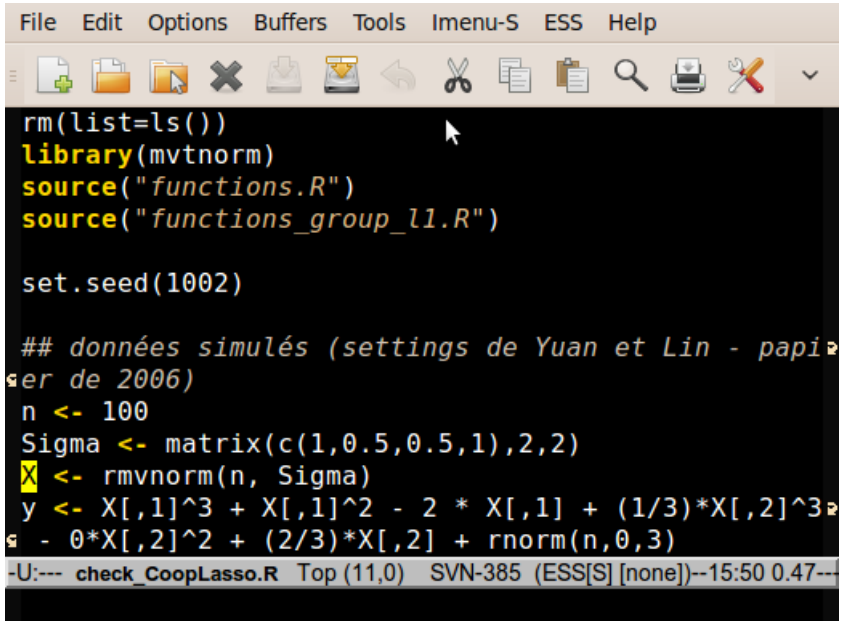
- Dans un répertoire `R`, placer
  - un répertoire `data`
  - un répertoire `mes_projets`
  - un répertoire `functions`
- Créer un répertoire par projet
  - sauvegarde des données

    ```
    save.image(file = "f.RData")
    ```
  - sauvegarde des instructions

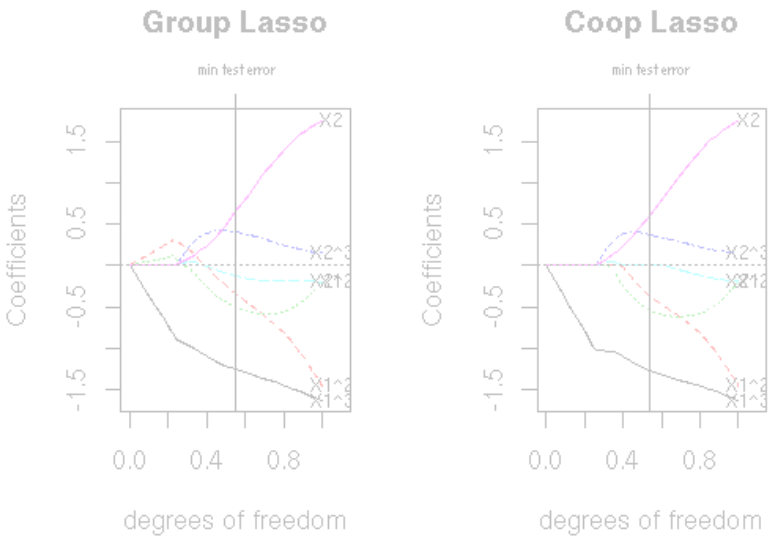
    ```
    savehistory(file = "f.Rhistory")
    ```
- `bibliotheques` contient les extensions installées.

Environnement de travail sous Linux

Un bureau de développement avec R

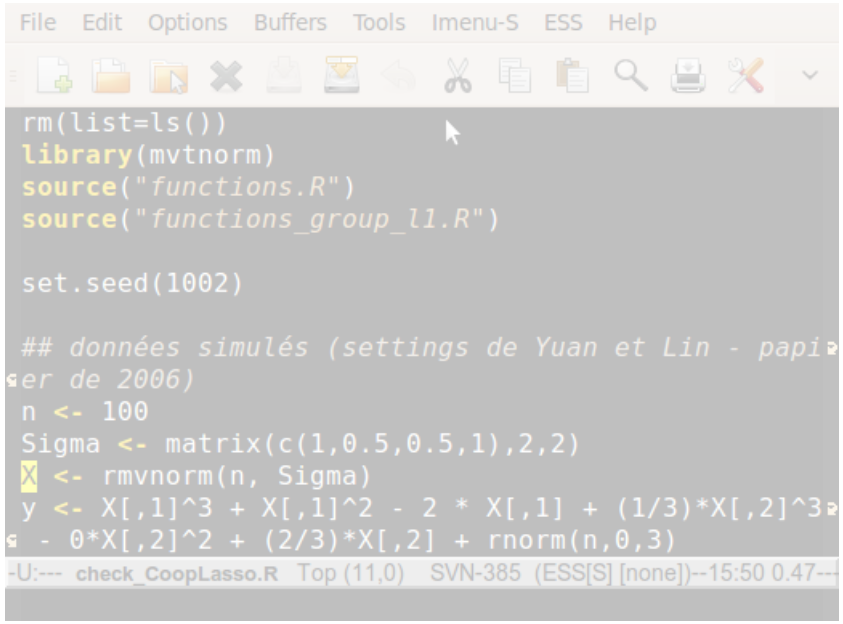


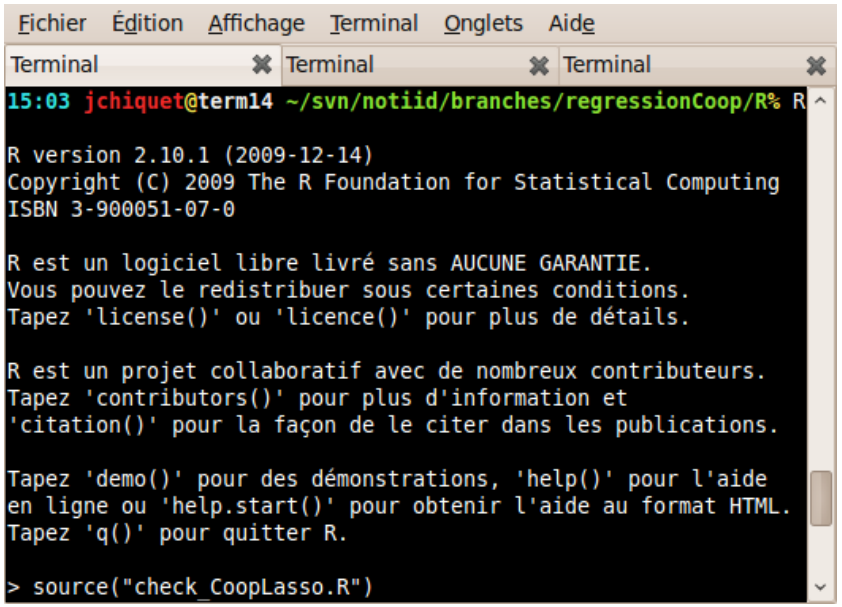
1. un éditeur de texte
2. un terminal avec R
3. des sorties graphiques



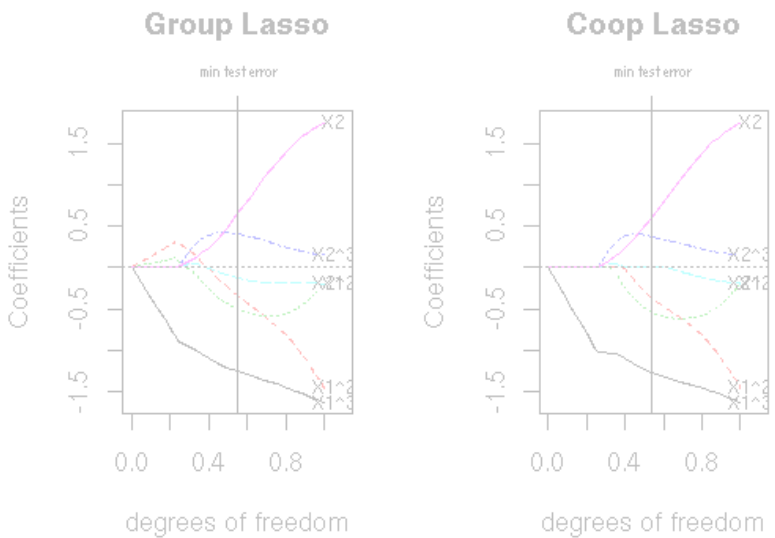
Environnement de travail sous Linux

Un bureau de développement avec R



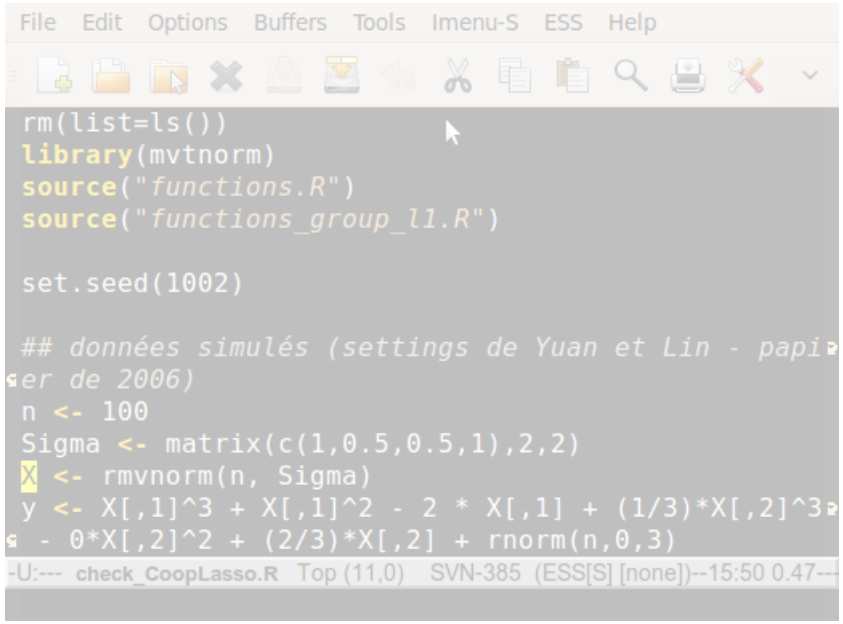


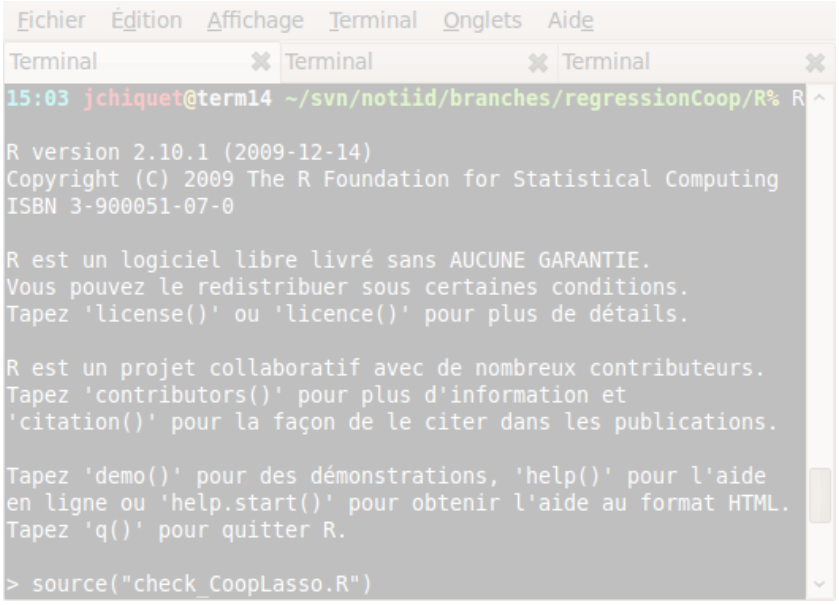
1. un éditeur de texte
2. un terminal avec R
3. des sorties graphiques



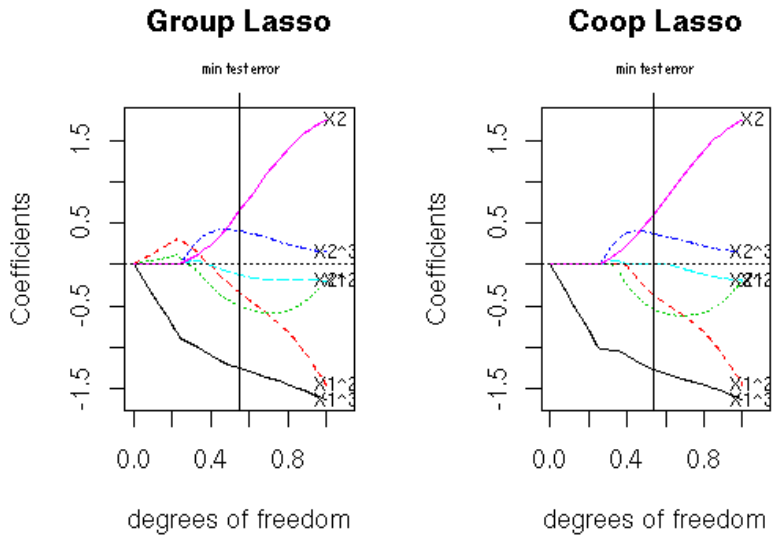
Environnement de travail sous Linux

Un bureau de développement avec R





1. un éditeur de texte
2. un terminal avec R
3. des sorties graphiques



Trouver de l'aide

Depuis R

- `help(str)` : lance l'aide associée à la commande `str`,
- `help.search("factorial")` : cherche les commandes contenant le mot-clé `factorial`,
- `help.start()` : lance l'aide HTML.

Sur le Web

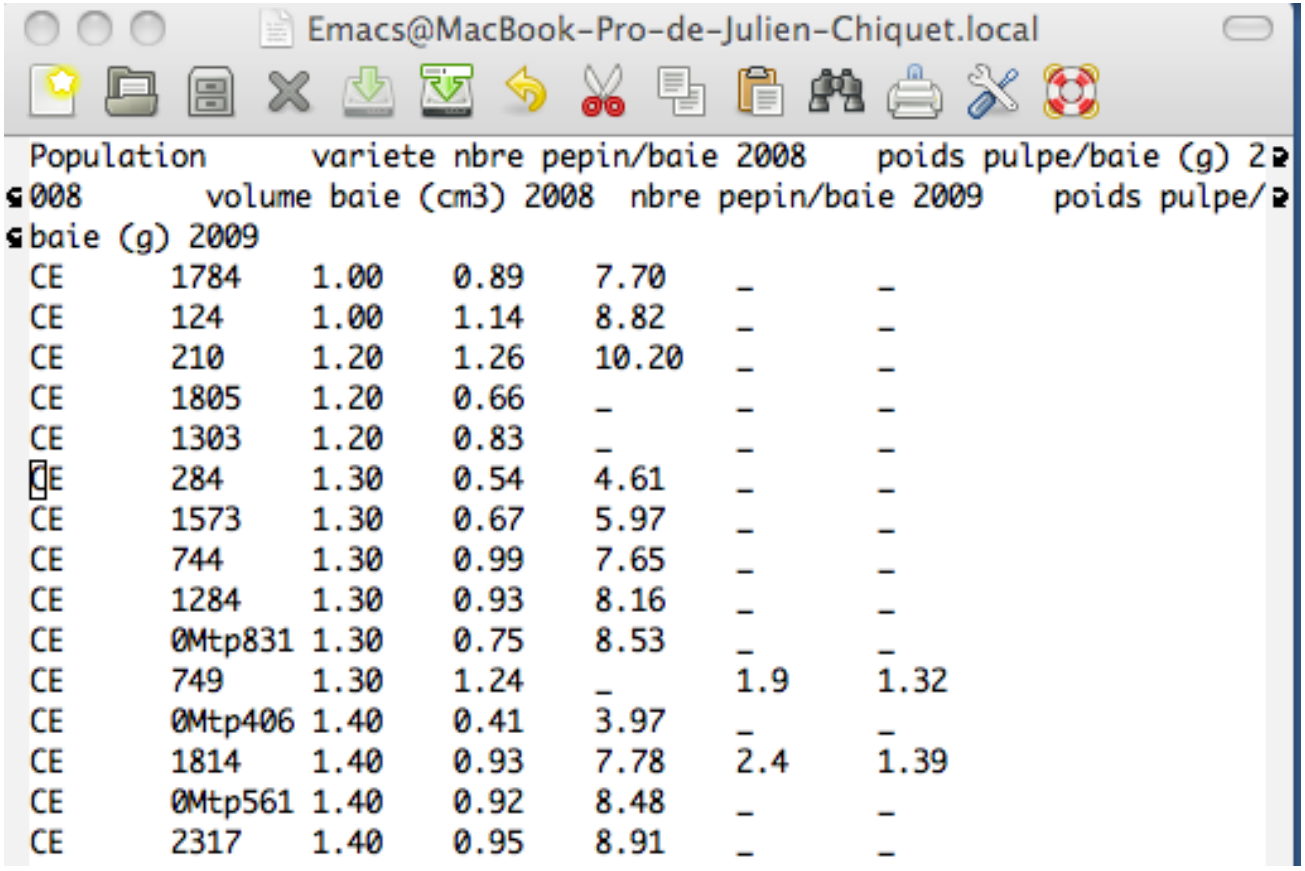
- Le site du CRAN : beaucoup (trop ?) de guides d'utilisations sont répertoriés (y compris ceux en français),
- le site du module en propose une sélection.

À tout moment

- la liste des commandes usuelles,
- le prof (pas infaillible mais rapide d'accès).

Analyse élémentaire d'un jeu de données

Quelle tête ont les données ? On ouvre avec Emacs :



Population	variete	nbre pepin/baie 2008	poids pulpe/baie (g) 2008	volume baie (cm3) 2008	nbre pepin/baie 2009	poids pulpe/baie (g) 2009
CE	1784	1.00	0.89	7.70	_	_
CE	124	1.00	1.14	8.82	_	_
CE	210	1.20	1.26	10.20	_	_
CE	1805	1.20	0.66	_	_	_
CE	1303	1.20	0.83	_	_	_
CE	284	1.30	0.54	4.61	_	_
CE	1573	1.30	0.67	5.97	_	_
CE	744	1.30	0.99	7.65	_	_
CE	1284	1.30	0.93	8.16	_	_
CE	0Mtp831	1.30	0.75	8.53	_	_
CE	749	1.30	1.24	_	1.9	1.32
CE	0Mtp406	1.40	0.41	3.97	_	_
CE	1814	1.40	0.93	7.78	2.4	1.39
CE	0Mtp561	1.40	0.92	8.48	_	_
CE	2317	1.40	0.95	8.91	_	_

FIGURE : données baies de vignes 2008/2009

Je remplace tous les _ par du vide (R le comprendra mieux) !

Rappatriement I

Les commandes `getwd()` et `setwd()` gèrent le répertoire de travail :

```
> setwd("~/SVN/gao/BaseStatBio/0_IntroductionBaseStat/")
> getwd()

[1] "/Users/tom/SVN/gao/BaseStatBio/0_IntroductionBaseStat"
```

Les données possèdent un entête et sont délimitées par des tabulations :

```
> donnees <- read.delim("mesures_baie_raisin_2008-2009.txt")
```

Qu'est-ce qui se trouve dorénavant dans mon itinéraire de recherche ?

```
> ls()

[1] "donnees"

> objects()

[1] "donnees"
```

Rappatriement II

Quelle tête ont mes données ?

```
> head(donnees)
```

```
  Population variete nbre.pepin.baie.2008 poids.pulpe.baie..g..2008
1         CE    1784                   1.0                       0.89
2         CE     124                   1.0                       1.14
3         CE     210                   1.2                       1.26
4         CE    1805                   1.2                       0.66
5         CE    1303                   1.2                       0.83
6         CE     284                   1.3                       0.54
  volume.baie..cm3..2008 nbre.pepin.baie.2009 poids.pulpe.baie..g..2009
1                   7.70                    NA                        NA
2                   8.82                    NA                        NA
3                  10.20                    NA                        NA
4                     NA                    NA                        NA
5                     NA                    NA                        NA
6                   4.61                    NA                        NA
```

Je les mets dans mon itinéraire de recherche

```
> attach(donnees)
```

Rappatriement III

```
Les objets suivants sont masqués from donnees (position 3):

    nbre.pepin.baie.2008, nbre.pepin.baie.2009,
    poids.pulpe.baie..g..2008, poids.pulpe.baie..g..2009, Population,
    variete, volume.baie..cm3..2008
Les objets suivants sont masqués from donnees (position 4):

    nbre.pepin.baie.2008, nbre.pepin.baie.2009,
    poids.pulpe.baie..g..2008, poids.pulpe.baie..g..2009, Population,
    variete, volume.baie..cm3..2008
```

Rappatriement IV

Et les attributs ?

```
> str(donnees)

'data.frame':        245 obs. of  7 variables:
 $ Population               : Factor w/ 3 levels "CE","CO","TE": 1 1 1 1 1 1 1 1 1
 $ variete                  : Factor w/ 245 levels "0Mtp1004","0Mtp1005",..: 113 66
 $ nbre.pepin.baie.2008     : num  1 1 1.2 1.2 1.2 1.3 1.3 1.3 1.3 1.3 ...
 $ poids.pulpe.baie..g..2008: num  0.89 1.14 1.26 0.66 0.83 0.54 0.67 0.99 0.93 0.7
 $ volume.baie..cm3..2008   : num  7.7 8.82 10.2 NA NA 4.61 5.97 7.65 8.16 8.53 ...
 $ nbre.pepin.baie.2009     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ poids.pulpe.baie..g..2009: num  NA NA NA NA NA NA NA NA NA NA ...
```

Troisième partie III

Structures de données

Plan
Structures de données

Plan
Structures de données

Vecteurs : définition

Propriétés

- objet le plus élémentaire sous R,
- collection d'entités de même nature,
- mode (ou type) défini par la nature des entités qui le composent.

Les modes possibles

1. numérique (`numeric`),
2. caractère (`caracter`),
3. logique (`boolean`).

Vecteurs : définition

Propriétés

- objet le plus élémentaire sous `R`,
- collection d'entités de même nature,
- mode (ou type) défini par la nature des entités qui le composent.

Les modes possibles

1. numérique (`numeric`),
2. caractère (`caracter`),
3. logique (`boolean`).

Vecteurs : définition

Propriétés

- objet le plus élémentaire sous `R`,
- collection d'entités de même nature,
- mode (ou type) défini par la nature des entités qui le composent.

Les modes possibles

1. numérique (`numeric`),
2. caractère (`caracter`),
3. logique (`boolean`).

Quelques vecteurs typés

1. Numérique

```
> x0 <- 0
> x1 <- c(-1,23,98.7)
> mode(x0)

[1] "numeric"
```

2. Caractère

```
> y0 <- "bonjour"
> y1 <- c("Pomme","Flore","Alexandre")
> mode(y1)

[1] "character"
```

3. Logique

```
> z0 <- TRUE
> z1 <- c(FALSE,TRUE,FALSE,TRUE,TRUE)
> z2 <- c(T,F,F)
> mode(z2)

[1] "logical"
```

Remarques sur l'affectation

Définition (affectation)

C'est l'opération qui consiste à attribuer une valeur à une variable.

En R, plusieurs choix sont possibles :

- l'opérateur usuel est '<-' (signe inférieur suivi du signe moins)

```
> jo <- "l'indien"
> jo
[1] "l'indien"
```

- l'opérateur '=' peut être utilisé la plupart du temps

```
> nb.max.d.annees.pour.faire.une.these = 3
> nb.max.d.annees.pour.faire.une.these
[1] 3
```

- la commande `assign` permet cette opération (d'où l'anglicisme *assignation*)

```
> assign("x", c(8,9,-pi,sqrt(2)))
> x
[1]  8.000000  9.000000 -3.141593  1.414214
```

Valeurs spéciales

Variables réservées par R

- `NA` est le code R pour les valeurs manquantes (absentes des données),
- `NaN` est le code de R pour signifier un résultat numérique aberrant ,
- `Inf` et `-Inf` sont les valeurs réservées pour plus et moins ∞,
- `NULL` est l'objet nul.

```
> c(4,2,NA,5)
[1]  4  2 NA  5
> 0/0
[1] NaN
> 1/0
[1] Inf
> names(1)
NULL
```

Opérations arithmétiques
s'effectuent terme-à-terme

Soient `x,y` tels que

```
> x<-c(1,2,-3,-4)
> y<-c(-5,-6,9,0)
```

- `'+'` addition des éléments de deux vecteurs

```
> x+y
[1] -4 -4  6 -4
```

- `'-'` soustraction des éléments de deux vecteurs

```
> x-y
[1]   6   8 -12  -4
```

- `'*'` multiplication des éléments de deux vecteurs

```
> x*y
[1]  -5 -12 -27   0
```

- `'/'` division des éléments de deux vecteurs

```
> x/y
[1] -0.2000000 -0.3333333 -0.3333333       -Inf
```

Le « recyclage » des éléments du vecteur

Lors d'une opération entre vecteurs, les vecteurs trop courts sont ajustés pour atteindre la taille du plus grand vecteur en recyclant les données.

Exemple

```
> x <- c(10,100,1000)
> y <- c(1,2)
> 2*x + y - 1
```

```
[1]   20  201 2000
```

$\rightsquigarrow$ souvent pratique mais attention aux effets de bords !

Opérateurs mathématiques I

Fonctions numériques élémentaires

`floor,ceiling,round`.

```
> floor(2/3)
[1] 0
> ceiling(2/3)
[1] 1
> round(2/3,3)
[1] 0.667
```

Fonctions arithmétiques élémentaires

`^,%%,%/%,abs,log,exp,log10,sqrt,cos,tan,sin...` s'appliquent toutes terme-à-terme.

```
> log10(c(10,100,1000))
```

Opérateurs mathématiques II

```
[1] 1 2 3
```

```
> cos(c(pi/2,pi))^2 + sin(c(pi/2,pi))^2
```

```
[1] 1 1
```

Fonctions caractérisant un vecteur

`prod, sum, max, min, range, which.min, which.max, length`

```
> x <- c(-8,1.5,3)
> prod(x)
```

```
[1] -36
```

```
> sum(x)
```

```
[1] -3.5
```

```
> length(x)
```

```
[1] 3
```

```
> max(x)
```

Opérateurs mathématiques III

```
[1] 3

> min(x)

[1] -8

> range(x)

[1] -8  3

> which.max(x)

[1] 3

> which.min(x)

[1] 1
```

Pour le minimum / maximum terme-à-terme : `pmin,pmax`.

Fonctions appliquées le long du vecteur

`cumsum, cumprod, cummin, cummax`

Opérateurs mathématiques IV

```
> x <- c(-2,1,-3,2)
> cumprod(x)

[1] -2 -2  6 12

> cumsum(x)

[1] -2 -1 -4 -2

> cummax(x)

[1] -2  1  1  2

> cummin(x)

[1] -2 -2 -3 -3
```

Opérateurs ensemblistes

Fonctionnent pour tous les modes

`unique,intersect,union,setdiff,setequal,is.element`

```
> unique(c("banane","citron","banane"))
[1] "banane" "citron"
> intersect(c("banane","citron"),c("orange","banane"))
[1] "banane"
> union(c("banane","citron"),c("orange","banane"))
[1] "banane" "citron" "orange"
> setequal(c("banane","citron"),c("orange","banane"))
[1] FALSE
> is.element(1,sample(c(1,2,3),2))
[1] TRUE
```

L'opérateur ' : '

`from:to`

Génère une séquence par pas de un depuis le nombre `from` jusqu'à `to` (si possible).

```
> -5:5
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
> 5:-5
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
> pi:6
[1] 3.141593 4.141593 5.141593
> 1:6/2
[1] 0.5 1.0 1.5 2.0 2.5 3.0
> 1:(6/2)
[1] 1 2 3
```

La commande seq

Plusieurs schémas possibles

- `seq(from,to)`
- `seq(from,to,by=)`
- `seq(from,to,length.out=)`

```
> seq(1,10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(2,10,by=2)
[1]  2  4  6  8 10
> seq(2,10,length.out=6)
[1]  2.0  3.6  5.2  6.8  8.4 10.0
```

La commande `rep`

Fonctionne pour tous les modes

- `rep(x,times)`, où `times` peut être un vecteur,
- `rep(x,each)`.

```
> rep(1,3)
[1] 1 1 1
> rep("Mercy",3)
[1] "Mercy" "Mercy" "Mercy"
> rep(c("A","B","C"),c(3,2,4))
[1] "A" "A" "A" "B" "B" "C" "C" "C" "C"
> rep(c(TRUE,FALSE), each=2)
[1]  TRUE  TRUE FALSE FALSE
```

Génération de vecteurs logiques

Obtenus par conditions avec

- les opérateurs logiques `'<'`, `'<='`, `'>'`, `'>='`, `'=='` `'!='`
- le ET, le OU, NON, OU exclusif : `'&'` (intersection), `'|'` (union), `'!'` (négation), `xor`.

```
> note1 <- c(8,9,14,3,17.5,11)
> note2 <- c("C","B","A","B","E","B")
> admis <- (note1 >= 10) & (note2 == "A" | note2 == "B")
> mention <- (note1 >= 15) & (note2 == "A")
> admis
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE
> sum(admis)
[1] 2
> sum(mention)
[1] 0
```

Par concaténation

Avec 'c()'

L'opérateur 'c()' peut s'appliquer à n'importe quoi pourvu que l'on concatène des vecteurs de même type.

```
> c( c(1,2), c(3,4))

[1] 1 2 3 4

> round(c(seq(-pi,pi,len=4),rep(c(1:3),each=2),0),2)

 [1] -3.14 -1.05  1.05  3.14  1.00  1.00  2.00  2.00  3.00  3.00  0.00
```

Remarque

Dans le second exemple, les entiers composants `c(1:3)` ont été forcés au typage flottant.

Par concaténation II

Avec `paste`

Concaténation de chaînes de caractères. Convertit en caractères les éléments passés en argument avant toute opération.

```
> paste("R","c'est","bien")

[1] "R c'est bien"

> paste(2:4,"ieme")

[1] "2 ieme" "3 ieme" "4 ieme"

> paste("A",1:5, sep="")

[1] "A1" "A2" "A3" "A4" "A5"

> paste("A",1:5, sep="",collapse="")

[1] "A1A2A3A4A5"
```

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Principe

- Permet la sélection d'un sous-ensemble du vecteur `x`.
- Le sous-ensemble est spécifié entre crochets `x[subset]`.

L'objet `subset` peut prendre 4 types différents :

1. un vecteur logique, qui doit être de la même taille que le vecteur `x` ;
2. un vecteur numérique aux composantes positives, qui spécifie les valeurs à inclure ;
3. un vecteur numérique aux composantes négatives, qui spécifie les valeurs à exclure ;
4. un vecteur de chaînes de caractères, qui spécifie les noms des éléments de `x` à conserver.

Indexation des vecteurs : exemples I

Vecteurs logiques

```
> x <- c(3,6,-2,9,NA,sin(-pi/6))
> x[x > 0]

[1]  3  6  9 NA

> x[!is.na(x)]

[1]  3.0  6.0 -2.0  9.0 -0.5

> x[!is.na(x) & x>0]

[1] 3 6 9

> mean(x,na.rm=TRUE)

[1] 3.1

> x[x <= mean(x,na.rm=TRUE)]

[1]  3.0 -2.0   NA -0.5
```

Indexation des vecteurs : exemples II

Vecteurs aux composantes positives ou négatives

```
> x
[1]  3.0  6.0 -2.0  9.0   NA -0.5
> x[2]
[1] 6
> x[1:5]
[1]  3  6 -2  9 NA
> x[-c(1,5)]
[1]  6.0 -2.0  9.0 -0.5
> x[-(1:5)]
[1] -0.5
```

Indexation des vecteurs : exemples III

Vecteurs de chaînes de caractères

```
> names(x) <- c("var1","var2","var3","var4","var5","var6")
> x

var1 var2 var3 var4 var5 var6
 3.0  6.0 -2.0  9.0   NA -0.5

> x[c("var1","var3")]

var1 var3
   3   -2
```

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,

2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;

3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,

2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;

3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Autres commandes d'indexation et de sélection

1. Classer
   - `sort` renvoie le vecteur classé par ordre croissant ou décroissant,
   - `order` renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,

2. Extraire
   - `which` renvoie les indices de `x` vérifiant une condition ;

3. Échantillonner
   - `sample` échantillonne aléatoirement dans un vecteur `x`, avec ou sans remise.

Exemples

```
> x <- -5:5
> y <- sample(x)
> sort(y)

 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5

> order(y)

 [1]  1  3 11  5  2 10  8  9  7  6  4

> y[order(y)]

 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5

> y[order(y,decreasing=TRUE)]

 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5

> which(sample(x,4) > 0)

[1] 3
```

Plan
Structures de données

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Définition

Définition

Un facteur est un vecteur de variables catégorielles. Les niveaux du facteur peuvent être ordonnés ou pas.

Utilisation

les facteurs s'utilisent pour catégoriser les données d'un vecteur (ce qui s'avère très utile pour la gestion des variables qualitatives).

$\rightsquigarrow$ un facteur est souvent associé à d'autres vecteurs pour en définir une partition.

Création : la fonction `factor`

```
> factor(sample(1:3,10,replace=TRUE))

 [1] 1 1 2 3 1 2 2 2 3 1
Levels: 1 2 3

> factor(sample(1:3,10,replace=TRUE),levels=1:5)

 [1] 2 2 1 3 2 3 3 1 1 2
Levels: 1 2 3 4 5
```

Gestion : `nlevels,levels,table`

```
> x <- factor(sample(c("thésard","CR","MdC"),15,replace=TRUE))
> cat(nlevels(x),"niveaux:",levels(x))

3 niveaux: CR MdC thésard

> table(x)

x
     CR     MdC thésard
      4       5       6
```

Un exemple de facteur associé à un vecteur

Données

Chacun me donne son âge et son grade [1]

```
> age <- c(25,35,32,27,32,40,26,25,26,28,30,NA,36,30,30)
> grd <- c("thd","CR","MdC","thd","thd","MdC","MdC","thd","thd","MdC","CR"
```

Question : nombre d'individus par catégorie ?

```
> table(grd)

grd
 CR MdC thd
  3   5   7
```

1. sauf un qui refuse :'(

Utilisation

Applique une fonction sur un vecteur partitionné en groupes.

Question : âge moyen / écart-type par catégorie ?

```
> tapply(age,grd,mean,na.rm=TRUE)

      CR       MdC       thd
33.66667 31.50000 27.85714

> tapply(age,grd,sd,na.rm=TRUE)

      CR       MdC       thd
3.214550 6.191392 2.794553
```

Plan
Structures de données

Plan
Structures de données

Tableau : définition

Définition (objet `array`)

Un tableau est un vecteur muni d'un attribut dimension (`dim`), lui même défini par un vecteur. Il est défini par la commande

`array(data,dim,dimnames=)`

```
> array(1:8,c(2,2,2))

, , 1

     [,1] [,2]
[1,]    1    3
[2,]    2    4

, , 2

     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Matrice : définition

Définition (objet `matrix`)

Une matrice est un tableau à deux dimensions. Elle est définie par la commande

`matrix(data,nrow=,ncol=,byrow)`

En conséquence

- Un objet `array` à deux dimensions est automatiquement converti en `matrix`
- Un vecteur auquel on ajoute un atribut dimension est automatiquement converti en `matrix`

```
> class(array(1:4,c(2,2)))

[1] "matrix"

> x <- c(1,2,3,4)
> dim(x) <- c(2,2)
> class(x)

[1] "matrix"
```

Remarques importantes

1. R range les éléments d'une matrice par défaut par colonne.

```
> matrix(1:6,nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> matrix(1:6,nrow=2,byrow=TRUE)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

2. Lors de la création d'une matrice, R recycle les éléments jusqu'à ce que les contraintes de dimension soient vérifiées.

```
> matrix(1:3,nrow=2,ncol=2)
     [,1] [,2]
[1,]    1    3
[2,]    2    1
```

Plan
Structures de données

Matrices : opérateurs élémentaires

Étant donné qu'une matrice est un vecteur pourvu d'une dimension, on a la proposition suivante :

Proposition

La plupart des opérateurs vectorielles s'appliquent (arithmétiques/mathématiques, ensemblistes, d'indexation).

```
> a <- matrix(sample(-4:4,9),3,3)
> cat(max(a),sum(a),prod(a))

4 0 0

> which(a > 0)

[1] 3 4 5 8

> cumsum(a[a > 0])

[1]  1  5  8 10

> order(a)

[1] 9 6 2 1 7 3 8 5 4

> round(exp(a),4)

       [,1]    [,2]   [,3]
[1,] 0.3679 54.5982 1.0000
[2,] 0.1353 20.0855 7.3891
[3,] 2.7183  0.0498 0.0183
```

Manipulation de matrices

Opérateurs matriciels usuels

- `+,/,*,^` sont les opérateurs usuels terme-à-terme,
- `%*%` est le produit matriciel,
- `crossprod()` est le produit scalaire,
- `t()` transpose une matrice,
- `diag()` extrait / spécifie la diagonale.

```
> a <- matrix(sample(-4:4,9),3,3)
> b <- matrix(sample(a),3,3)
> diag(a)

[1]  4 -1  0

> diag(a) <- diag(b) <- 1
> diag(a)

[1] 1 1 1

> a + t(b) %*% b

     [,1] [,2] [,3]
[1,]   15  -10   -5
[2,]   -5   34   -1
[3,]    1   -5   12
```

Concaténation de matrices

Trois fonctions selon l'effet voulu :

1. `c()` concatène les éléments de plusieurs matrices en un vecteur,
2. `cbind()` empile horizontalement plusieurs matrices,
3. `rbind()` empile verticalement plusieurs matrices.

```
> a <- matrix(1,2,3)
> b <- matrix(2,2,3)
> c(a,b)

 [1] 1 1 1 1 1 1 2 2 2 2 2 2

> cbind(a,b)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    2    2    2
[2,]    1    1    1    2    2    2

> rbind(a,b)

     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    2    2    2
[4,]    2    2    2
```

Résolution de systèmes linéaires, inversion matricielle

La commande `solve` résout

$$\mathbf{A}x = \mathbf{b},$$

```
> A <- matrix(c(4,2,8,-3),2,2)
> b <- c(2,3)
> solve(A,b)
```

```
[1]  1.0714286 -0.2857143
```

ou inverse une matrice :

```
> round(solve(A) %*% A,8)
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Commandes avancées d'algèbre linéaire
Utile pour l'analyse numérique

`R` dispose des outils classiques d'algèbre linéaire

- `det` : calcule le déterminant d'une matrice ;
- `chol` : factorisation de Cholesky ($A = C^\intercal C$, avec A symétrique, C triangulaire supérieure) ;
- `qr` : factorisation QR ($A = QR$ avec Q orthogonale, R triangulaire supérieure) ;
- `eigen` : calcule valeurs propres et vecteurs propres d'une matrice ;
- `svd` : calcule la décomposition en valeurs singulières.
- `...`

Plan
Structures de données

Liste : définition

Définition (objet `list`)

Une liste est une collection d'objets hétérogènes. Elle est définie par la commande `list(el1=, el2=, ...)`. Les éléments d'une liste peuvent posséder un nom.

```
> list(c(1,2,3),c("robert","johnson"),matrix(rnorm(4),2,2))
[[1]]
[1] 1 2 3

[[2]]
[1] "robert"  "johnson"

[[3]]
          [,1]       [,2]
[1,] 1.3934589  0.4581159
[2,] 0.7934216 -0.8848944
> list(numero = c(1,2,3), noms = c("robert","johnson"), mat = matrix(rnorm(4),2,2))
$numero
[1] 1 2 3

$noms
[1] "robert"  "johnson"

$mat
          [,1]      [,2]
[1,] -1.925150 0.5999631
[2,] -1.764114 0.1307384
```

Accéder aux éléments

Deux situations

1. Les éléments de la liste ne sont pas nommés : on accède au i^{e} élément par indexation `nom_liste[[i]]` uniquement.
2. Les éléments de la liste sont nommés : on peut y accéder comme ci-dessus ou en utilisant le nom de l'élément `nom_liste$nom_elt`.

```
> maliste <- list(numero = c(1,2,3), noms = c("robert","johnson"), mat = matrix(rnor
> maliste$nom
[1] "robert"  "johnson"
> maliste$nom[2]
[1] "johnson"
> maliste[[2]]
[1] "robert"  "johnson"
> maliste[[2]][2]
[1] "johnson"
```

Manipulation de listes I

Sélectionner des éléments

Fonctionne (presque) comme pour les vecteurs

```
> l1 <- list(1:2,c("a","c","g","t"))
> l1[[-2]]

[1] 1 2
```

Commande `lapply`

Applique une fonction à chaque élément d'une liste

```
> lapply(maliste,length)

$numero
[1] 3

$noms
[1] 2

$mat
[1] 4
```

Manipulation de listes II

Commande `c()`

Permet de concaténer deux listes.

```
> c(list(1:2,c("a","c","g","t")),list(rnorm(3),"yop"))

[[1]]
[1] 1 2

[[2]]
[1] "a" "c" "g" "t"

[[3]]
[1]  0.6874990  1.2186725 -0.9696582

[[4]]
[1] "yop"
```

Définition (objet `data.frame`)

C'est une liste à laquelle on impose certaines contraintes[2]*, afin de rassembler vecteurs et facteurs sous la forme d'un tableau de données.*

- Pratiquement, *un tableau de données est une matrice dont les colonnes sont de mode différent,*
- C'est l'objet idéal pour la manipulation de données (forcez-vous à l'utiliser).

2. que je vous épargne

Définition (objet `data.frame`)

C'est une liste à laquelle on impose certaines contraintes[2]*, afin de rassembler vecteurs et facteurs sous la forme d'un tableau de données.*

- Pratiquement, *un tableau de données est une matrice dont les colonnes sont de mode différent,*
- C'est l'objet idéal pour la manipulation de données (forcez-vous à l'utiliser).

2. que je vous épargne

Création de tableau de données

Syntaxe

On peut spécifier le nom des colonnes par le vecteur `row.names` ou directement comme pour une liste :

`data.frame(e1=,e2=,...,row.names=)`

```
> age <- c(25,35,32,27,32,40,26,25,26,28,30,NA,36,30,30)
> grd <- c("thd","CR","MdC","thd","thd","MdC","MdC","thd","thd","MdC","CR","MdC","CR
> sex <- factor(sample(c(rep("M",3),rep("F",12))))
> donnees <- data.frame(age=age,grade=grd,sexe=sex)
> head(donnees)
```

```
  age grade sexe
1  25   thd    F
2  35    CR    F
3  32   MdC    F
4  27   thd    F
5  32   thd    M
6  40   MdC    F
```

Manipulation des éléments du tableau de données

- Comme une liste !
- les commandes `attach()` / `detach` placent / ôtent les éléments du tableaux de données dans l'itinéraire de recherche.

```
> donnees$age

 [1] 25 35 32 27 32 40 26 25 26 28 30 NA 36 30 30

> attach(donnees,warn.conflicts=FALSE)
> grade

 [1] thd CR  MdC thd thd MdC MdC thd thd MdC CR  MdC CR  thd thd
Levels: CR MdC thd

> detach(donnees)
```

Travailler avec les tableaux de données

- beaucoup de fonctions prédéfinies
- penser aux fonctions `tapply` (ou `by`)

```
> summary(donnees)
```

```
      age        grade   sexe
 Min.   :25.00   CR :3   F:12
 1st Qu.:26.25   MdC:5   M: 3
 Median :30.00   thd:7
 Mean   :30.14
 3rd Qu.:32.00
 Max.   :40.00
 NA's   :1
```

```
> attach(donnees,warn.conflicts=FALSE)
> by(age,sexe,mean,na.rm=TRUE)
```

```
sexe: F
[1] 29.45455
------------------------------------------------------------
sexe: M
[1] 32.66667
```

```
> by(age,grade,mean,na.rm=TRUE)
```

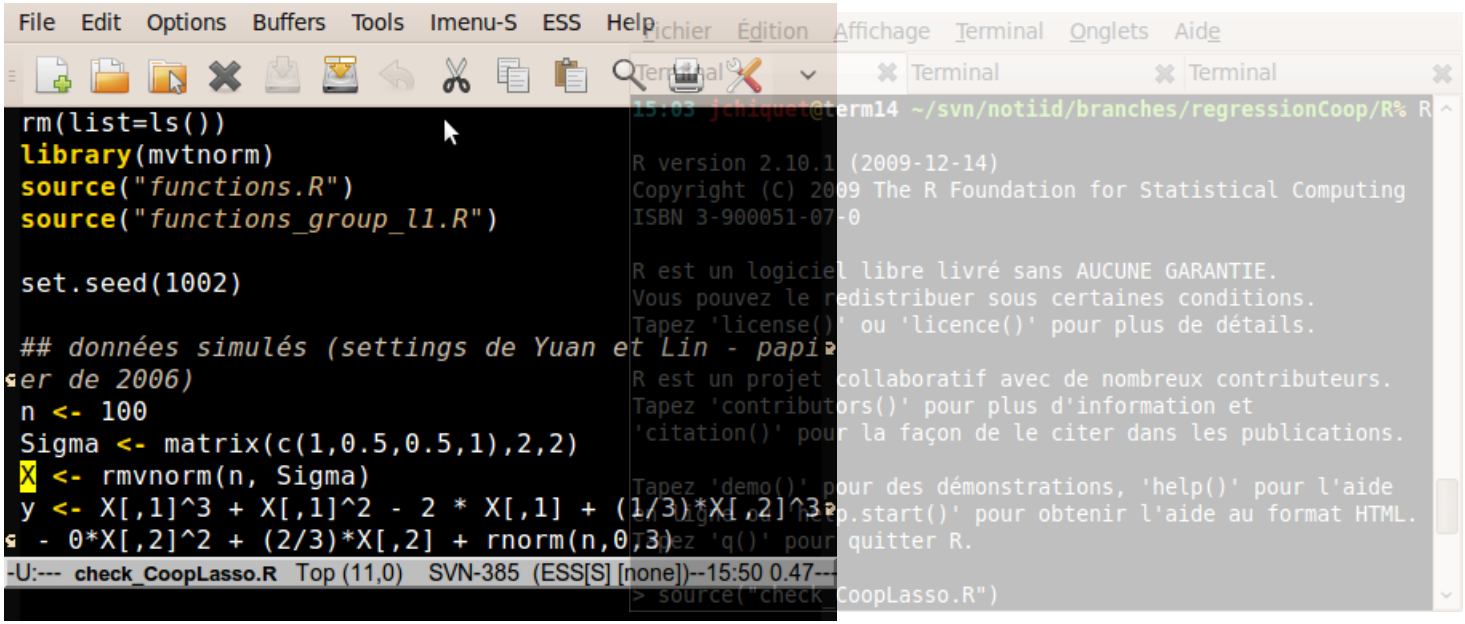
```
grade: CR
[1] 33.66667
------------------------------------------------------------
grade: MdC
[1] 31.5
------------------------------------------------------------
grade: thd
[1] 27.85714
```

```
> detach(donnees)
```

Quatrième partie IV

Développer avec R

Programmer en pratique : rappels



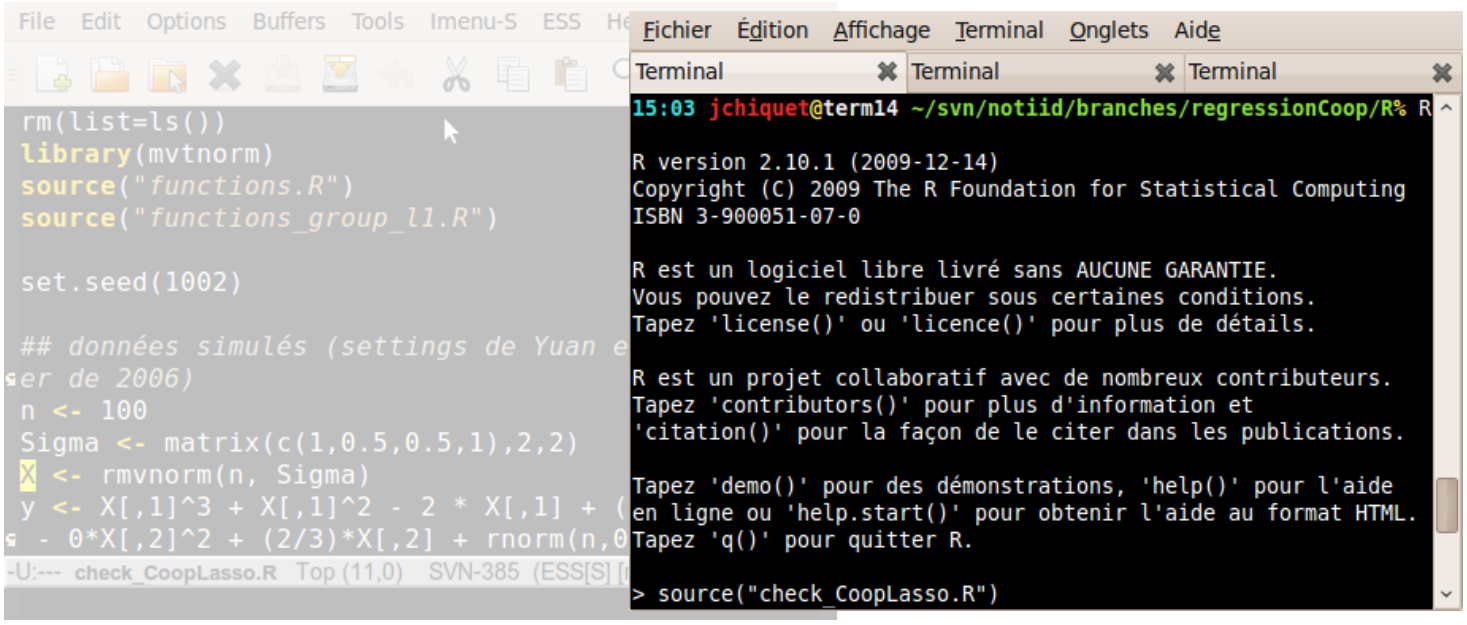
1. un éditeur de texte (vos fonctions / scripts)
2. un terminal avec R (tester, « sourcer »)

« Sourcer »

`source("un_script.R")` : exécute la série de commandes de `mon_script.R`

`source("mes_fonctions.R")` : charge le contenu (les fonctions) de `mes_fonctions.R`

Programmer en pratique : rappels



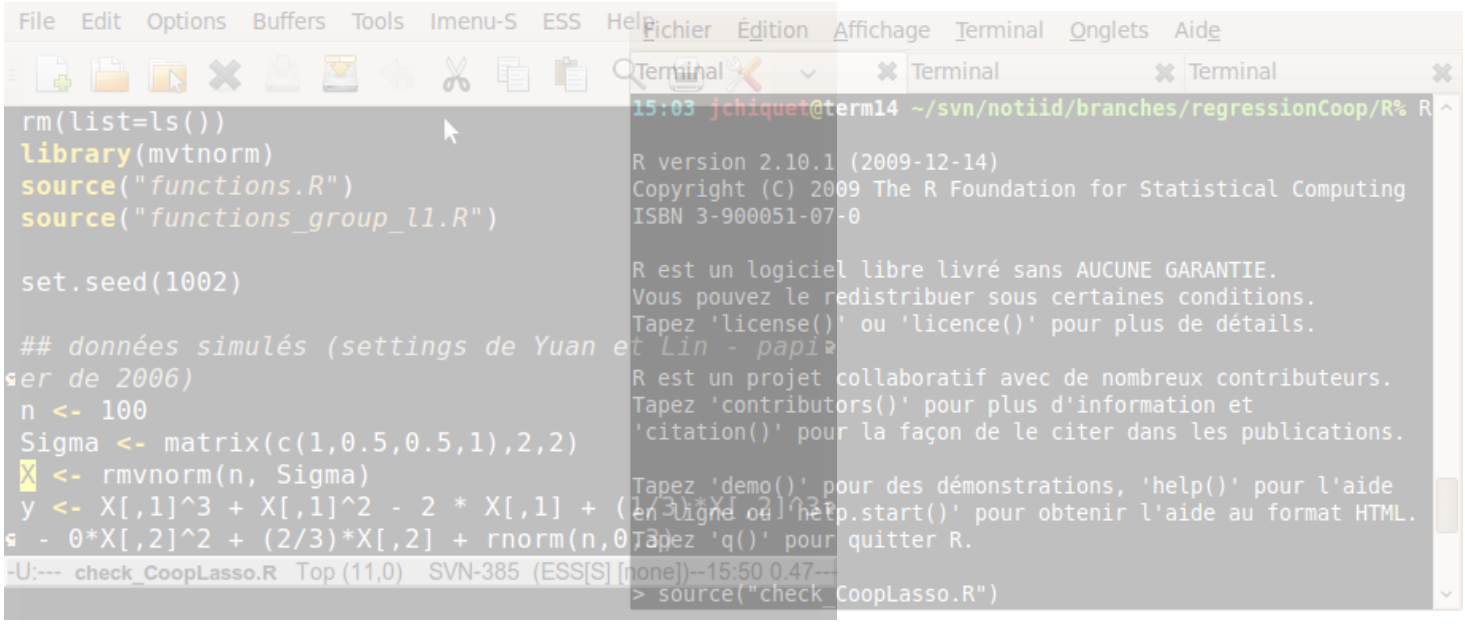
1. un éditeur de texte (vos fonctions / scripts)
2. un terminal avec R (tester, « sourcer »)

« Sourcer »

`source("un_script.R")` : exécute la série de commandes de `mon_script.R`

`source("mes_fonctions.R")` : charge le contenu (les fonctions) de `mes_fonctions.R`

Programmer en pratique : rappels



1. un éditeur de texte (vos fonctions / scripts)
2. un terminal avec `R` (tester, « sourcer »)

« Sourcer »

`source("un_script.R")` : exécute la série de commandes de `mon_script.R`

`source("mes_fonctions.R")` : charge le contenu (les fonctions) de `mes_fonctions.R`

Regrouper les expressions

Syntaxe

```
{expr_1; expr_2; ...; expr_n }
```

ou

```
{
  expr_1
   ...
  expr_n
}
```

Remarques sur les groupes

- La dernière valeur du groupe est retournée ;
- un groupe d'expressions peut être passé à une fonction, réutilisé dans une expression plus grande, etc.

Exécution conditionnelle : `if`, `if/else`, `ifelse`

Syntaxe

```
if (condition) {
  expr_1
} else {
  expr_2
}
```

ou

```
ifelse(condition, a, b)
```

Remarques

- `condition` est une valeur logique : penser à `&`, `|`, `!`, ...
- le `else` est optionnel,
- `elseif` permet d'imbriquer les conditionnements.

Exécution répétée : boucle `for`

Syntaxe

```
for (var in set) {
  expr(var)
}
```

ou

```
for (var in set)
  expr(var)
```

à fuir pour éviter les effets de bords sournois !

Remarques sur la boucle `for`

- `var` est la variable incrémentée,
- `set` est un vecteur définissant les valeurs successives,
- *lente* comparée aux opérateurs matriciels.

Exécution répétée : boucles `while` et `repeat`

Syntaxe

```
while (condition) {
  expr
}
```

ou

```
repeat {
  expr
}
```

Remarque

- Comme pour `for`, éviter les imbrications sources de lenteur.

Contrôle des boucles : `break, next`

Exemples d'utilisation

```
repeat {
  expr
  if (condition) {break}
}
```

ou

```
while (condition1){
  expr_1
  if (condition2) {next}
  expr_2
}
```

Remarque

- `break` est la seule manière d'interrompre une boucle `repeat`.

Définir une fonction

Syntaxe

```
nom_de_la_fonction <- function(arg1,arg2, ...) {
  expression

  return(var)
}
```

Remarques

- `return` peut être omis (à éviter) : dans ce cas la dernière valeur calculée est renvoyée.
- peut être tapée directement dans l'interpréteur ou dans un fichier externe `functions.R`, chargé par `source`.

Un exemple simple

Moyenne empirique d'un vecteur

Avec suppression des valeurs manquantes !

```
> moyenne <- function(x) {
+   ## suppression des valeurs manquantes
+   x.not.na <- x[!is.na(x)]
+   ## moyenne empirique
+   resultat <- sum(x.not.na) / length(x.not.na)
+
+   return(resultat)
+ }
```

Tests

```
> moyenne(rnorm(100))
```

```
[1] 0.04183799
```

```
> moyenne(c(1,-5,3,NA,8.7))
```

```
[1] 1.925
```

Les arguments, leurs valeurs par défaut
Encore un point fort de R

Propriétés

- *les arguments peuvent être passés dans le désordre s'ils sont nommés :* `var=object,`
- *on peut définir une valeur par défaut pour n'importe quel argument lors de la définition de la fonction :* `var=10`.
- *en cas de sorties multiples, les sorties doivent être renvoyées sous forme de liste.*

Remarques

- *Les valeurs par défaut rendent la lecture des fonctions beaucoup plus aisée pour l'utilisateur : imposer peu d'arguments obligatoires.*
- *Les noms des éléments de la liste définie dans la fonction sont conservés à l'extérieur de la fonction.*

Les arguments, leurs valeurs par défaut
Encore un point fort de R

Propriétés

- *les arguments peuvent être passés dans le désordre s'ils sont nommés :* `var=object`,
- *on peut définir une valeur par défaut pour n'importe quel argument lors de la définition de la fonction :* `var=10`.
- *en cas de sorties multiples, les sorties doivent être renvoyées sous forme de liste.*

Remarques

- *Les valeurs par défaut rendent la lecture des fonctions beaucoup plus aisée pour l'utilisateur : imposer peu d'arguments obligatoires.*
- *Les noms des éléments de la liste définie dans la fonction sont conservés à l'extérieur de la fonction.*

Les arguments, leurs valeurs par défaut
Encore un point fort de R

Propriétés

- *les arguments peuvent être passés dans le désordre s'ils sont nommés :* `var=object`,
- *on peut définir une valeur par défaut pour n'importe quel argument lors de la définition de la fonction :* `var=10`.
- *en cas de sorties multiples, les sorties doivent être renvoyées sous forme de liste.*

Remarques

- *Les valeurs par défaut rendent la lecture des fonctions beaucoup plus aisée pour l'utilisateur : imposer peu d'arguments obligatoires.*
- *Les noms des éléments de la liste définie dans la fonction sont conservés à l'extérieur de la fonction.*

Les arguments, leurs valeurs par défaut
Encore un point fort de R

Propriétés

- *les arguments peuvent être passés dans le désordre s'ils sont nommés : `var=object`,*
- *on peut définir une valeur par défaut pour n'importe quel argument lors de la définition de la fonction : `var=10`.*
- *en cas de sorties multiples, les sorties doivent être renvoyées sous forme de liste.*

Remarques

- *Les valeurs par défaut rendent la lecture des fonctions beaucoup plus aisée pour l'utilisateur : imposer peu d'arguments obligatoires.*
- *Les noms des éléments de la liste définie dans la fonction sont conservés à l'extérieur de la fonction.*

Un exemple (un tout petit peu) plus avancé

Résumé numérique d'un vecteur

```
> resume <- function(x,na.rm=TRUE,affiche=FALSE) {
+   mu <- mean(x,na.rm=na.rm)
+   s2 <- var(x,na.rm=na.rm)
+   if (affiche) {
+     cat("\nMoyenne:",mu,"et variance:",s2)
+   }
+   return(list(moyenne = mu, variance = s2))
+ }


> out <- resume(rnorm(100))
> out$variance

[1] 0.861812

> out <- resume(affiche=TRUE,x=rexp(100,0.5))

Moyenne: 2.060797 et variance: 3.115734
```

Motivations : reproductibilité de la recherche

Principe de Claerbout (Géophysicien, Stanford)

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

Démarche

1. Proposer une méthode et exposer dans un article ses propriétés,
2. Écrire et déposer un package sur CRAN,
3. Publier un article dans « journal of statistical software » ou une note dans « Bioinformatics ».

Motivations : reproductibilité de la recherche

Principe de Claerbout (Géophysicien, Stanford)

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.

Démarche

1. Proposer une méthode et exposer dans un article ses propriétés,
2. Écrire et déposer un package sur CRAN,
3. Publier un article dans « journal of statistical software » ou une note dans « Bioinformatics ».

Simplicité de la création d'un package

Définir un objectif

Par exemple, `SIMoNe` : construire un graphe des interactions significatives entre gènes à partir de données du transcriptome.

Organisation type

1. Fichier DESCRIPTION
2. Répertoire R : fonctions R (fonction `inferGraph(data)`)
3. Répertoire man : documentation des fonctions
4. Répertoire data : données
5. (Répertoire src : pour les fichiers à compiler, header etc.)

Motivations

Produire un document scientifique, c'est

- expérimenter

  (pipette + éprouvette + blouse) ou ordi $\rightsquigarrow$ données
- analyser les résultats

  méthode + logiciel + données $\rightsquigarrow$ graphes
- rédiger des observations

  idée + (bloc-note ou traitement de texte) $\Rightarrow$ texte
- mettre en forme

  texte + graphes + traitement de texte $\Rightarrow$ article

Motivations

Produire un document scientifique, c'est

- expérimenter

  (pipette + éprouvette + blouse) ou ordi $\rightsquigarrow$ données

- analyser les résultats

  méthode + logiciel + données $\rightsquigarrow$ graphes

- rédiger des observations

  idée + (bloc-note ou traitement de texte) $\Rightarrow$ texte

- mettre en forme

  texte + graphes + traitement de texte $\Rightarrow$ article ☺

Motivations

Produire un document scientifique, c'est

- expérimenter

  (pipette + éprouvette + blouse) ou ordi $\rightsquigarrow$ données

- analyser les résultats

  méthode + logiciel + données $\rightsquigarrow$ graphes

- rédiger des observations

  idée + (bloc-note ou traitement de texte) $\Rightarrow$ texte

- mettre en forme

  texte + graphes + traitement de texte $\Rightarrow$ article

Motivations

Produire un document scientifique, c'est

- expérimenter

  (pipette + éprouvette + blouse) ou ordi $\rightsquigarrow$ données

- analyser les résultats

  méthode + logiciel + données $\rightsquigarrow$ graphes

- rédiger des observations

  idée + (bloc-note ou traitement de texte) $\Rightarrow$ texte

- mettre en forme

  texte + graphes + traitement de texte $\Rightarrow$ article

Motivations

Produire un document scientifique, c'est

- expérimenter

  (pipette + éprouvette + blouse) ou ordi $\rightsquigarrow$ données

- analyser les résultats

  méthode + logiciel + données $\rightsquigarrow$ graphes

- rédiger des observations

  idée + (bloc-note ou traitement de texte) $\Rightarrow$ texte

- mettre en forme

  texte + graphes + traitement de texte $\Rightarrow$ article 

Solutions

1.
   - analyser : MS Excel
   - rédiger : MS Word
   - mettre en forme : MS Word
2.
   - analyser : MatLab
   - rédiger : OpenOffice
   - mettre en forme : OpenOffice
3.
   - analyser : R
   - rédiger : Emacs
   - mettre en forme : LaTeX

mon opinion [a] : 

a. de geek *très* subjective

Le package `sweave` de LaTeX permet de faire appel à du code R dans le document

Solutions

1.
   - analyser : MS Excel
   - rédiger : MS Word
   - mettre en forme : MS Word
2.
   - analyser : MatLab
   - rédiger : OpenOffice
   - mettre en forme : OpenOffice
3.
   - analyser : R
   - rédiger : Emacs
   - mettre en forme : LaTeX

mon opinion [a] : ☹

---

a. de geek *très* subjective

Le package `sweave` de LaTeX permet de faire appel à du code R dans le document

Solutions

1.
   - analyser : MS Excel
   - rédiger : MS Word
   - mettre en forme : MS Word
2.
   - analyser : MatLab
   - rédiger : OpenOffice
   - mettre en forme : OpenOffice
3.
   - analyser : R
   - rédiger : Emacs
   - mettre en forme : LaTeX

mon opinion [a] : ☺

a. de geek *très* subjective

Le package `sweave` de LaTeX permet de faire appel à du code R dans le document

Solutions

1.
   - analyser : MS Excel
   - rédiger : MS Word
   - mettre en forme : MS Word
2.
   - analyser : MatLab
   - rédiger : OpenOffice
   - mettre en forme : OpenOffice
3.
   - analyser : R
   - rédiger : Emacs
   - mettre en forme : LaTeX

mon opinion [a] : ☺

a. de geek *très* subjective

Le package `sweave` de LaTeX permet de faire appel à du code R dans le document

Code latex

```
\documentclass[a4paper]{article}
\begin{document}
In this example we embed parts of the examples
\texttt{kruskal.test} help page into a \LaTeX{} document: ...
\end{document}
```

Sortie pdf

In this example we embed parts of the examples from the `kruskal.test` help page into a LaTeX document : ...

Le module Sweave I

Fichier mélange ".rnw"

```
\documentclass[a4paper]{article}
\begin{document}
In this example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX\ document:

<<>>=

data(airquality)
kruskal.test(Ozone  ~ Month, data = airquality)

@

which shows that the location parameter of the Ozone
distribution varies significantly from month to month.
Finally we include a boxplot of the data:

<<fig=TRUE,echo=FALSE>>=

boxplot(Ozone  ~ Month, data = airquality)

@

\end{document}
```

Le module Sweave II

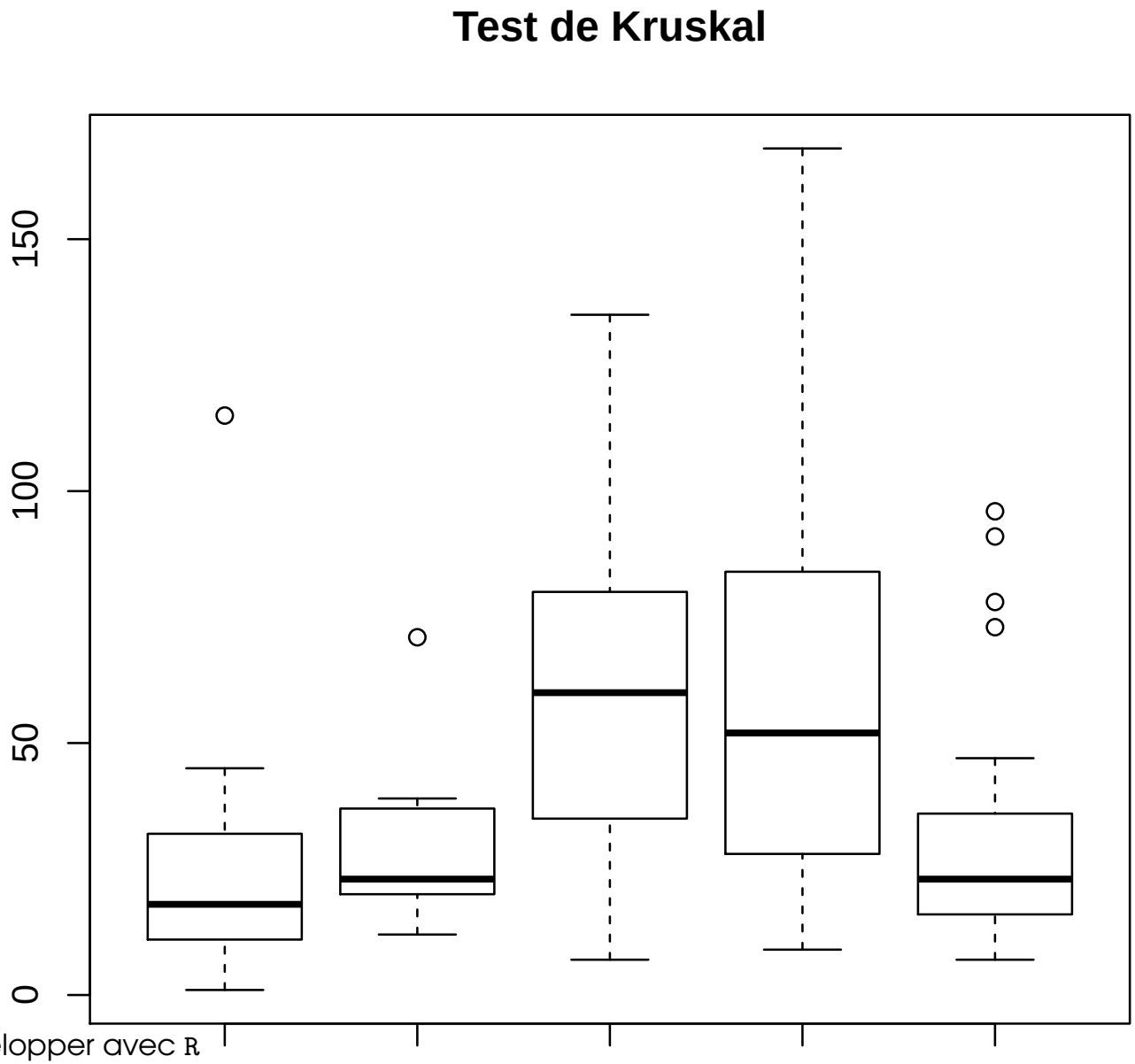
Fichier LaTeX ".tex"

```
\documentclass[a4paper]{article}
\usepackage{Sweave}
\begin{document}
In this example we embed parts of the examples from the
\texttt{kruskal.test} help page into a \LaTeX\ document:
\begin{Sinput}
  R> data(airquality)
  R> kruskal.test(Ozone ~ Month, data = airquality)
\end{Sinput}
\begin{Soutput}
  Kruskal-Wallis rank sum test data: Ozone by Month Kruskal-Wallis
  chi-squared = 29.2666, df = 4, p-value = 6.901e-06
\end{Soutput}
which shows that the location parameter of the Ozone
distribution varies significantly from month to month.
Finally we include a boxplot of the data:
\includegraphics{example-1-002}
\end{document}
```

In this example we embed parts of the examples from the `kruskal.test` help page into a LaTeX document :

```
> data(airquality)
> kruskal.test(Ozone ~ Month, data = airquality)

        Kruskal-Wallis rank sum test

data:  Ozone by Month
Kruskal-Wallis chi-squared = 29.2666, df = 4, p-value = 6.901
```

which shows that the location parameter of the Ozone distribution varies significantly from month to month. Finally we include a boxplot of the data :


Test de Kruskal
0
50
100
150

Cinquième partie V

Entrées / Sorties

commande `scan`

Une utilisation élémentaire de `scan` permet une saisie plus agréable que la saisie directe des éléments d'un vecteur.

```
> x<-scan()
1: 1
2: 2
3: 3
4: 4
5: 5
6:
Read 5 items
>
> x
[1] 1 2 3 4 5
```

$\rightsquigarrow$ valable pour les jeux de données d'au plus quelques dizaines d'éléments...

Éditer des données

commande `edit`

Permet d'éditer des données existantes à l'aide d'un mini-tableur. Utile pour faire de petites modifications.

```
> new.data <- edit(old.data)
```

FIGURE : Éditeur Mac OS 10.6 / R 2.10

commandes `save` et `load`

La commande `save` permet de sauvegarder un sous ensemble des données de l'espace de travail dans un fichier binaire ; `load` permet de les recharger.

```
> x <- rnorm(125)
> y <- 1 + x + x^2
> save(file="mes_simus",x,y)
> rm(list=ls())
> objects()

character(0)

> load(file="mes_simus")
> objects()

[1] "x" "y"
```

Jeux de données prédéfinies

commande `data`

R dispose d'une collection de données prédéfinies directement utilisables. La commande `data()` permet de les lister puis de les charger.

```
> data(iris3)
> head(iris3)
```

```
[1] 5.1 4.9 4.7 4.6 5.0 5.4
```

- La description d'un jeu de données est accessible dans l'aide.
- L'installation d'un nouveau package rend souvent disponibles de nouveaux jeux de données accessibles par `data`.

D'abord, un bon éditeur . . . il vous permet de constater le formatage d'un fichier texte et comment en « attaquer » l'importation.

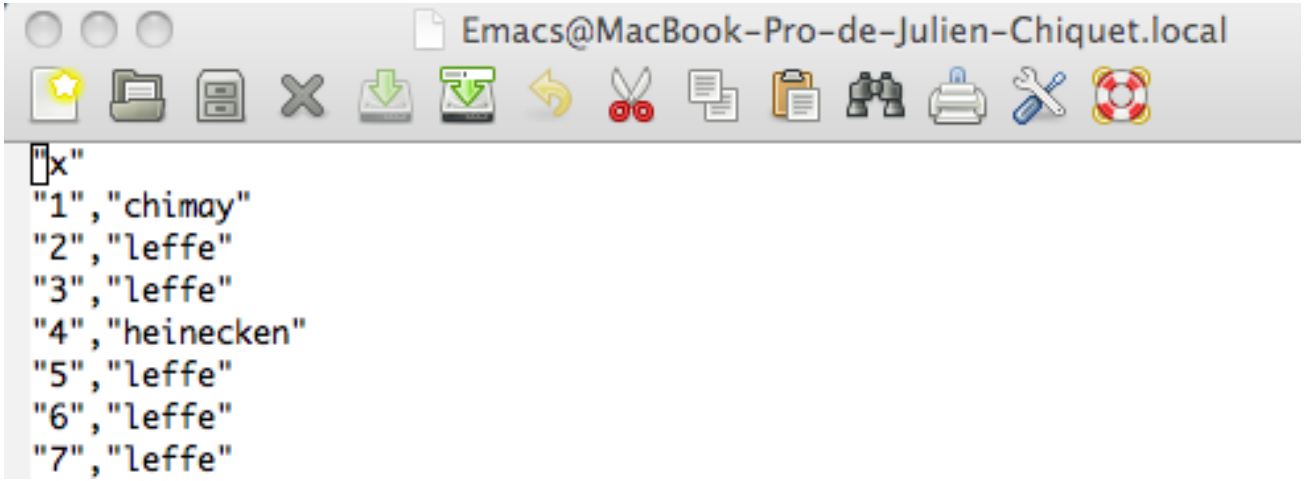


FIGURE : Fichier au formatage "csv"

commande `read.table`

Elle permet de lire un fichier formaté sous forme de table.
`read.table` stocke les données sous forme d'objet `data.frame`.

```
> mes_donnees<-read.table("mesures_baie_raisin_2008-2009.txt",header=TRUE,sep="\t")
> head(mes_donnees)
  Population variete nbre.pepin.baie.2008 poids.pulpe.baie..g..2008
1         CE    1784                   1.0                       0.89
2         CE     124                   1.0                       1.14
3         CE     210                   1.2                       1.26
4         CE    1805                   1.2                       0.66
5         CE    1303                   1.2                       0.83
6         CE     284                   1.3                       0.54
  volume.baie..cm3..2008 nbre.pepin.baie.2009 poids.pulpe.baie..g..2009
1                   7.70                    NA                       NA
2                   8.82                    NA                       NA
3                  10.20                    NA                       NA
4                     NA                    NA                       NA
5                     NA                    NA                       NA
6                   4.61                    NA                       NA
```

Lecture de fichiers III

`read.csv,read.delim`

commandes `read.csv` et `read.delim`

Ce sont des raccourcis pour la fonction `read.table`, spécialisés dans l'importation des données « .csv » (*comma-separated value*) ou tabulées (le séparateur est la tabulation).

commandes `write.table`, `write.csv` et `write.delim`

La fonction `write.table` permet d'imprimer les données issues d'une `data.frame` dans un fichier texte externe. `write.csv` et `write.delim` sont des raccourcis pour les données `csv` ou tabulée.

Pour aller plus loin. . .

Beaucoup de choses sur l'importation des données dans

R Data Import /Export.
`http://cran.r-project.org/doc/manuals/R-data.pdf`

- Exemples avancés avec `read.table`,
- communication avec les bases de données (SQL),
- importation de données Excel,
- . . .

Paramètres récurrents

Forme générique

La plupart des fonctions graphique s'utilisent par un appel du type

1. `nom.function(object, options)`,
2. `nom.function(x, y , options)`.

Parmi les options les plus courantes, on trouve :

- type="p" ; spécifie le type de tracé : "p" pour points, "l" pour lignes, "b" pour points liés par des lignes, "o" pour lignes superposées aux points. . .
- xlim= ; ylim=, spécifie les limites de axes x et y
- xlab= ; ylab=, annotation des axes x et y
- main= ; titre du graphe en cours
- sub= ; sous-titre du graphe en cours
- add=FALSE ; si TRUE superpose le graphe au précédent
- axes=TRUE ; si FALSE ne trace pas d'axes

Paramètres récurrents

Forme générique

La plupart des fonctions graphique s'utilisent par un appel du type

1. `nom.function(object, options)`,
2. `nom.function(x, y , options)`.

Parmi les options les plus courantes, on trouve :

- type="p" ; spécifie le type de tracé : "p" pour points, "l" pour lignes, "b" pour points liés par des lignes, "o" pour lignes superposées aux points...
- xlim= ; ylim=, spécifie les limites de axes x et y
- xlab= ; ylab=, annotation des axes x et y
- main= ; titre du graphe en cours
- sub= ; sous-titre du graphe en cours
- add=FALSE ; si TRUE superpose le graphe au précédent
- axes=TRUE ; si FALSE ne trace pas d'axes

Tracer une fonction numérique I

commande `plot`

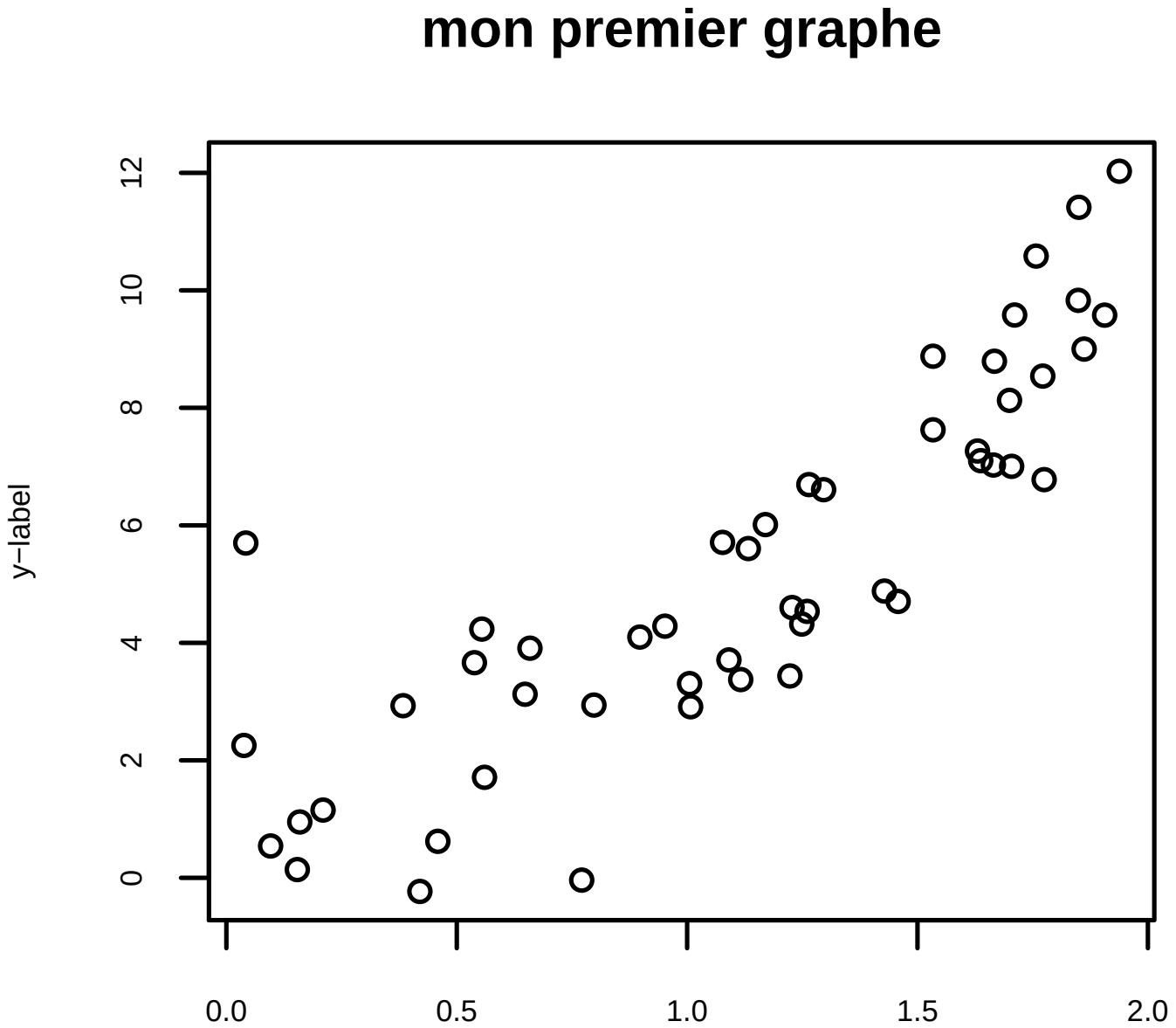
Fonction élémentaire de réprésentation graphique.

- `plot(vect)` représente le graphe des valeurs de `vect` sur l'axe des y.
- `plot(vect1,vect1)` représente le graphe des valeurs de `vect2` en fonction de `vect1`.

Par exemple, avec deux vecteurs :

```
> x <- runif(50,0,2)
> y <- 3 * x + 2 * x^2 + 1 + rnorm(50,sd=1.5)
> plot(x, y, xlab="x-label",ylab="y-label",main="mon premier graphe")
```

Tracer une fonction numérique II



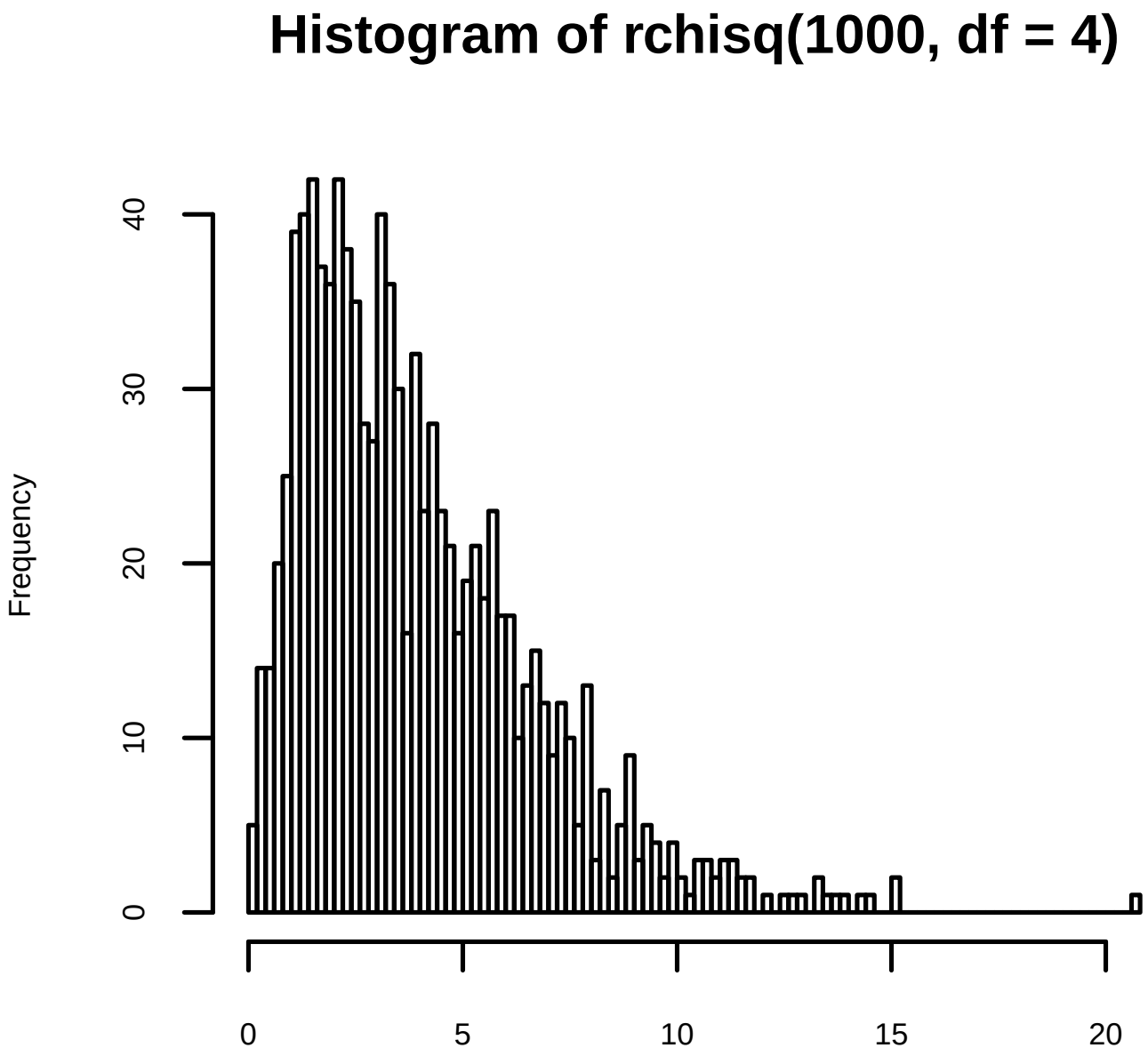
Tracer un objet R I
Toujours avec la commande plot

Beaucoup d'objet R accepte la commande plot ! En particulier, les histogrammes :

```
> mon_histo <- hist(rchisq(1000,df=4),nclass=75)
> plot(mon_histo,main="distribution empirique du Khi-2")
```

Tracer un objet R II

Toujours avec la commande `plot`



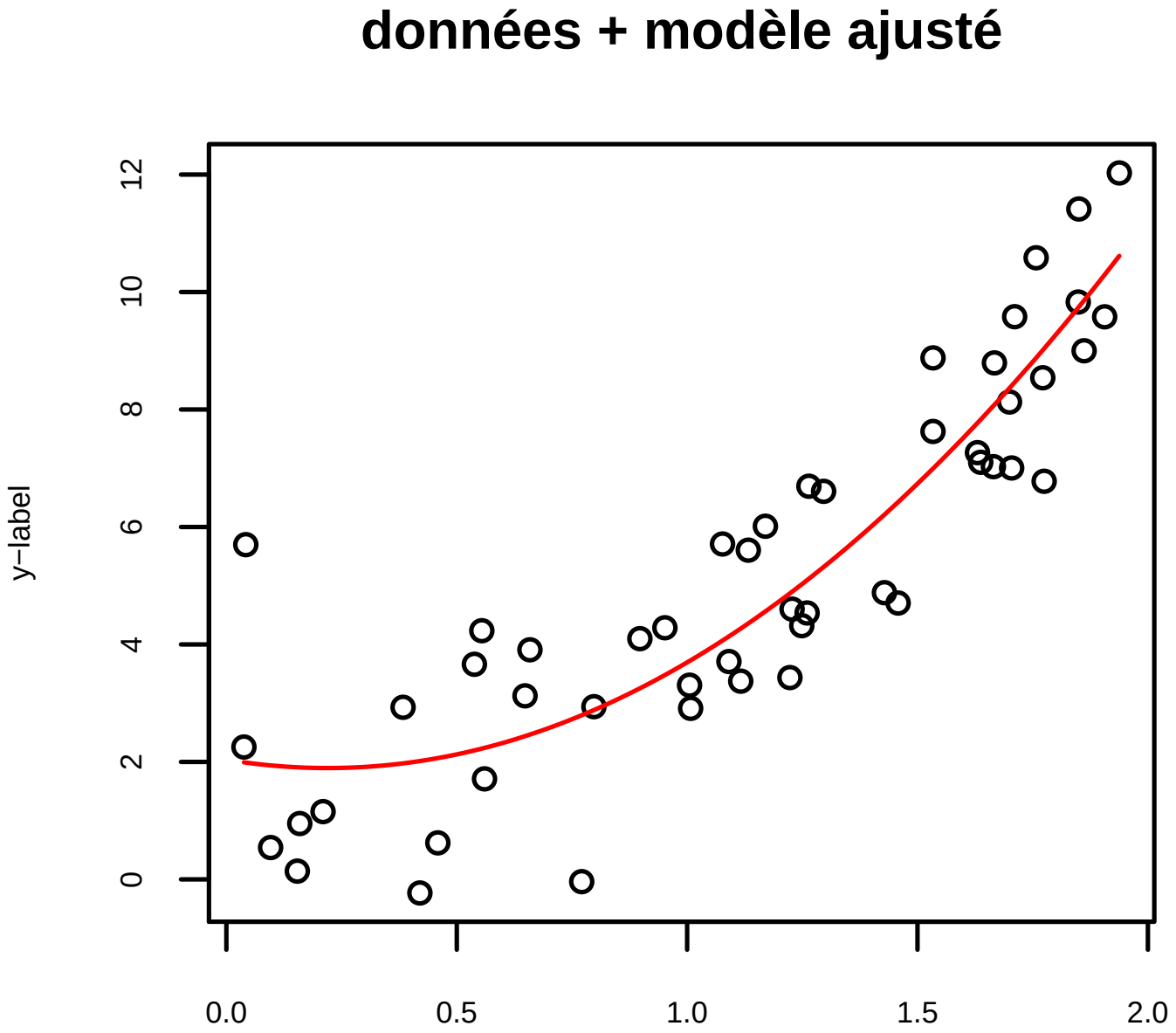
Tracer une fonction symbolique I

commande `curve`

Elle permet de tracer une fonction définie par une expression de `x`.

```
> plot(x, y, main="données + modèle ajusté",
+      xlab="x-label", ylab="y-label")
> a <- coefficients(lm(y~1+x+I(x^2)))
> curve(a[1] + a[2]*x + a[3]*x^2,add=TRUE,col="red")
```

Tracer une fonction symbolique II



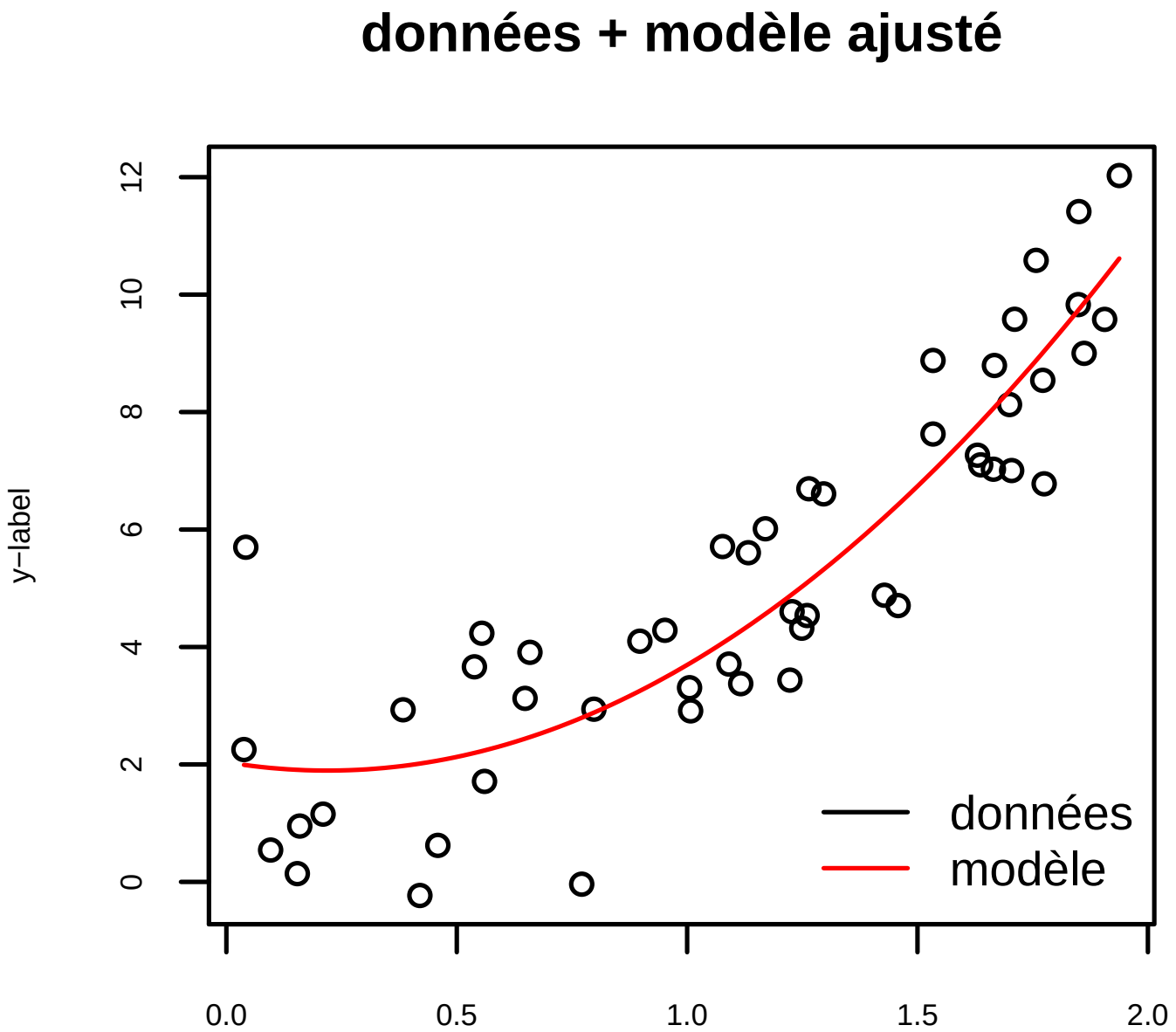
Ajouter une légende I

commande `legend`

Pour ajouter une légende. Attention aux options, assez nombreuses !

```
> plot(x, y, main="données + modèle ajusté",
+      xlab="x-label", ylab="y-label")
> a <- coefficients(lm(y~1+x+I(x^2)))
> curve(a[1] + a[2]*x + a[3]*x^2,add=TRUE,col="red")
>  legend("bottomright",c("données","modèle"),lty=c(1,1),col=c("black","red"),bty="n
```

Ajouter une légende II



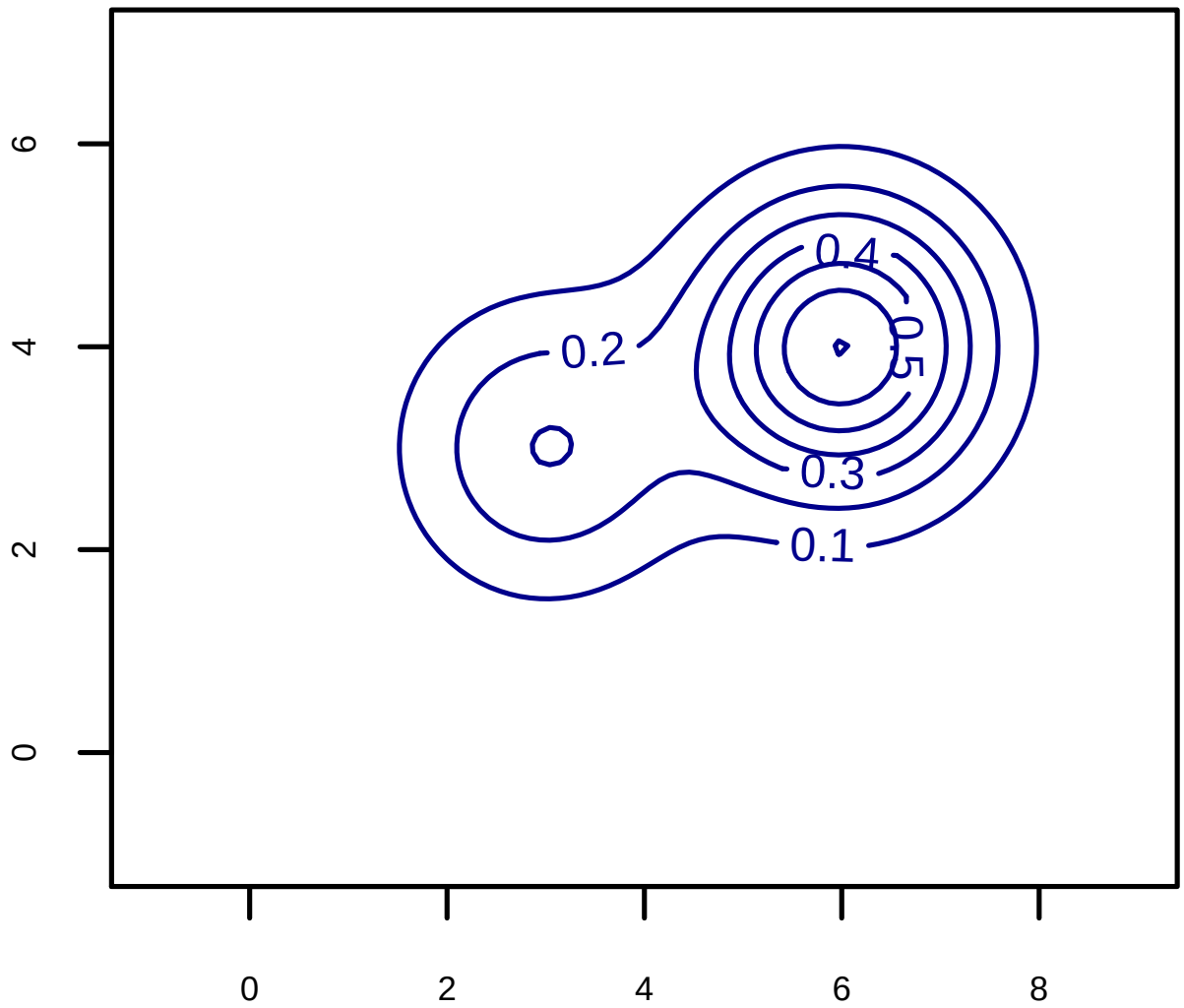
Représentation 3D (courbe de niveaux) I

commande `contour`

`contour(x,y,z)` permet de tracer des courbes de niveaux : `x` et `y` sont des vecteurs et `z` une matrice telle que les dimensions de `z` soient `length(x),length(y)`.

```
> x<-seq(-1,9,length=100)
> y<-seq(-1,7,length=100)
> z<-outer(x,y,function(x,y) 0.3*exp(-0.5*((x-3)^2 +(y -3)^2)) +
+          0.7*exp(-0.5*((x-6)^2 +(y -4)^2)))
> contour(x,y,z,col="blue4")
```

Représentation 3D (courbe de niveaux) II



Ajout de droites I

commande `abline`

`abline` permet d'ajouter à un graphe courant

- des droites de décalage `a` et de coefficient directeur `b` avec `abline(a,b)`,
- des droites verticales avec `abline(v=)`,
- des droites horizontales avec `abline(h=)`.

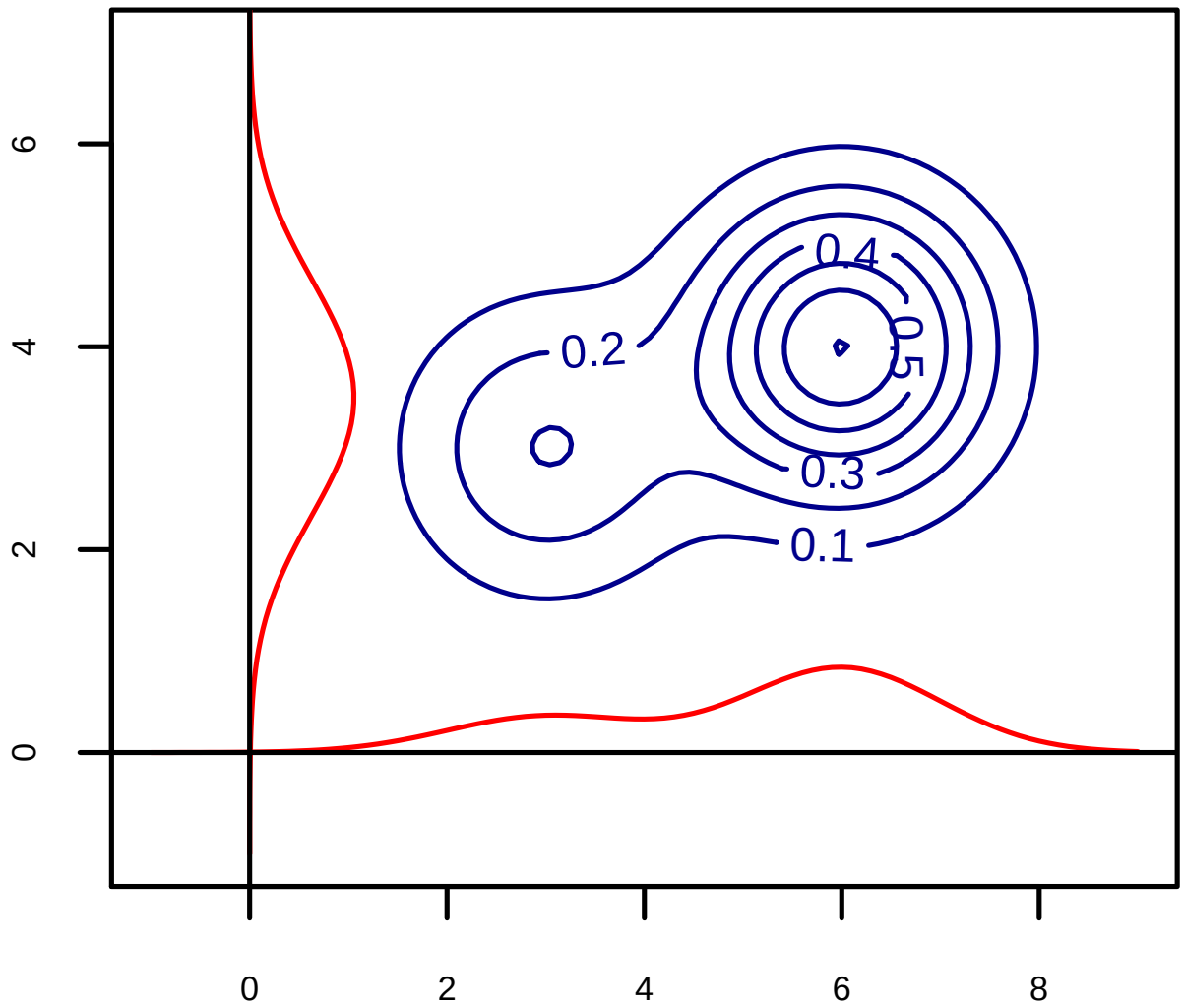
commandes `lines` et `points`

Pour ajouter une courbe ou des points : s'utilisent de manière similaire à `plot`.

Ajout de droites II

```
> contour(x,y,z,col="blue4")
> curve((0.3*dnorm(x,mean=3) + 0.7*dnorm(x,mean=6))*3,-1,9,col="red",ylim=
> x<-seq(-1,9,length=100)
> lines((0.5*dnorm(x,mean=3) + 0.5*dnorm(x,mean=4))*3,x,col="red")
> abline(h=0)
> abline(v=0)
```

Ajout de droites III



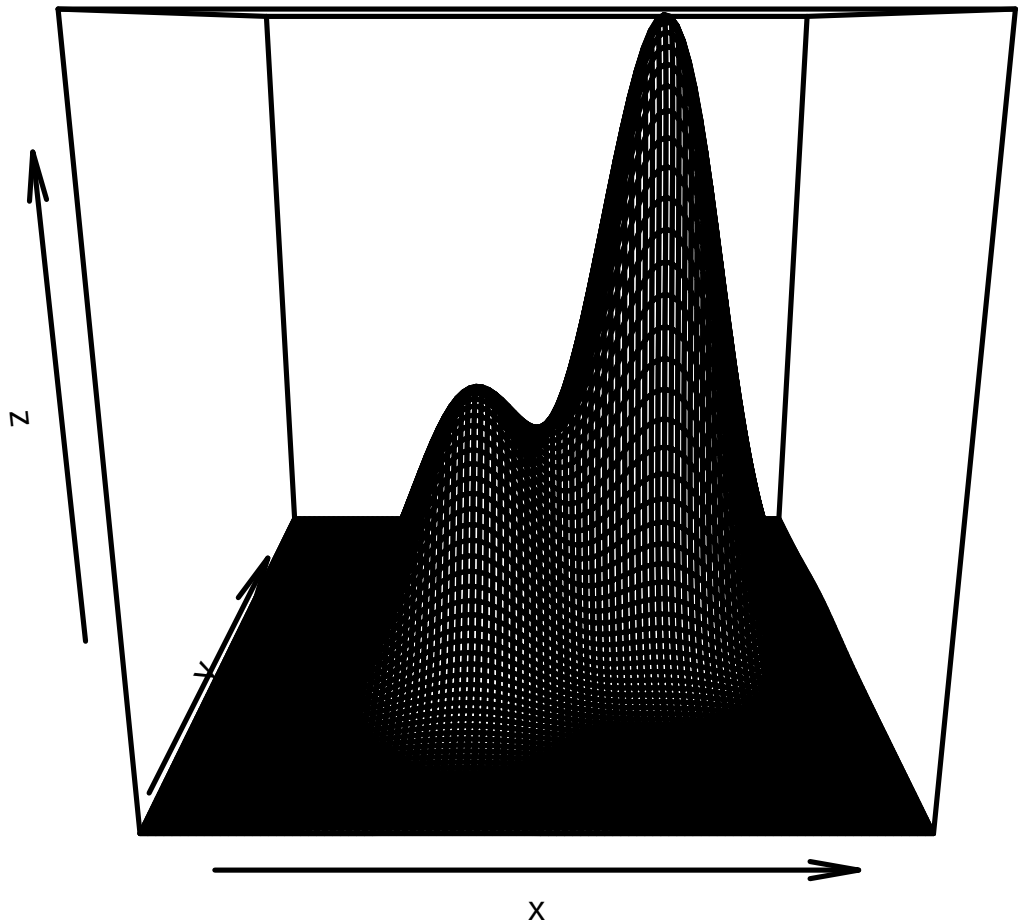
Graphe en 3D I

commande `persp`

Fonctionne comme la fonction `contour` en proposant une représentation en perpective.

```
> persp(x,y,z)
```

Graphe en 3D II



Rediriger la sortie graphique

Par défaut, R envoie les graphiques sur la sortie *écran*. De nombreuses

Exportation de graphes

Se réalise en encadrant les fonctions graphiques par les commandes `format_export(file="nom_fichier")` et `dev.off()`, où `format_fichier` peut prendre les valeurs `pdf`,`postscrip`,`png`,....

```
pdf(file="ma\_sortie.pdf")
plot(runif(20),runif(20))
dev.off()
```

Graphes multiples I

Ouverture d'une nouvelle fenêtre graphique

Se fait, selon les plateformes, avec les commandes

- `x11()` pour Linux,
- `quartz()` ou `x11()` pour Mac OS,
- `windows()`.

Découpage d'une fenêtre

Plusieurs possibilités :

- `layout(mat,width=,height=)`, qui s'utilise en découpant l'écran via la matrice `mat`.
- `par(mfrow=vect)` ou `par(mfcol=vect)` qui découpent en n lignes et m colonne spécifiées par le vecteur `vect`. Le remplissage se fait par ligne ou par colonne selon la fonction choisie.

Pour aller plus loin

- D'autres fonctions de haut niveau dans la partie dédiée aux statistiques
- Utiliser la liste des commandes usuelles pour les options et fonctions secondaires,
- La commande `par` gère les options graphiques,
- Consulter le package `lattice`, *extrêmement* puissant.

Lattice : Multivariate Data Visualization with R
Deepayan Sarkar
http://lmdvr.r-forge.r-project.org/

$\rightsquigarrow$ Cette page web propose toutes les figures et tous les codes R correspondant à leur génération !

Sixième partie VI

Statistiques descriptives

Variable quantitative discrète ou qualitative ordinale

Variable qualitative nominale

Variable continue

Représentations multivariées

Variable quantitative discrète ou qualitative ordinale

Variable qualitative nominale

Variable continue

- Résumés numériques
- Tableau de fréquences
- Fonction de répartition empirique
- Histogramme et estimateur à noyaux
- Boite à moustache

Représentations multivariées

- Description bidimensionnelle
- Description multidimensionnelle

Observations

Nous considérerons dans un premier temps une seule colonne du tableau de données, soient n observations :

$$x_1, \ldots, x_n$$

Variable quantitative discrète ou qualitative ordinale

La variable prend ses valeurs dans

$$V_x = \{\epsilon_1, ..., \epsilon_K\}$$

avec

$$\epsilon_1 < ... < \epsilon_K$$

Tableau de fréquence

- ϵ_k, la modalité
- n_k, l'effectif des observations ayant la valeur ϵ_k
- $f_k = \frac{n_k}{n}$, la fréquence
- $F_k = \sum_{j=1} kf_j$, la fréquece relative cumulée

Variable qualitative nominale

Même représentation sans l'ordre.
Pas de fréquence cumulée.

Camembert et diagramme en barres

Couplés à la commande `table` Le diagramme en barres et le graphe en camembert permettent de visualiser le découpage d'une population en donnée catégorielle.

```
> vigne <- read.delim("mesures_baie_raisin_2008-2009.txt",header=TRUE)
> vigne <- vigne[-c(2,6,7)]
> colnames(vigne) <- c("pop","pepin.08","poids.08","volcm3.08")
> head(vigne)
```

```
  pop pepin.08 poids.08 volcm3.08
1  CE      1.0     0.89      7.70
2  CE      1.0     1.14      8.82
3  CE      1.2     1.26     10.20
4  CE      1.2     0.66        NA
5  CE      1.2     0.83        NA
6  CE      1.3     0.54      4.61
```

```
> attach(vigne)
> par(mfrow=c(1,2))
> pie(table(pop))
> barplot(table(pop),las=3)
```

Tendance centrale

- Moyenne : $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$
  - Remarque : la somme des écarts à la moyenne empirique est nulle
$$\sum_i (x_i - \bar{x}) = 0$$
  - Inconvénient : problème des valeurs aberrantes
- Moyenne tronquée : $M_k = \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_{(i)}$ où $x_{(i)}$ est l'observation de rang i
- Médiane :
$$M = \begin{cases} x_{(n/2)} \text{ si } n \text{ est pair,} \\ x_{(\lfloor n/2 \rfloor + 1)} \text{ sinon} \end{cases}$$
- Fractile empirique d'ordre α
$$\hat{f}_\alpha = \begin{cases} x_{(n\alpha)} \text{ si } n\alpha \text{ est entier,} \\ x_{(\lfloor n\alpha \rfloor + 1)} \text{ sinon} \end{cases}$$

Indicateurs de dispersion

- la variance empirique
- la variance empirique corrigéee
- l'étendue
- l'étendue interquartile

Reprenons l'exemple de la vigne

Renommons les variables et considérons uniquement les données de 2008 pour une manipulation plus agréable. J'enlève également la colonne « variété », car je ne vois pas à quoi elle sert.

```
> vigne <- read.delim("mesures_baie_raisin_2008-2009.txt",header=TRUE)
> vigne <- vigne[-c(2,6,7)]
> colnames(vigne) <- c("pop","pepin.08","poids.08","volcm3.08")
> head(vigne)

  pop pepin.08 poids.08 volcm3.08
1  CE      1.0     0.89      7.70
2  CE      1.0     1.14      8.82
3  CE      1.2     1.26     10.20
4  CE      1.2     0.66        NA
5  CE      1.2     0.83        NA
6  CE      1.3     0.54      4.61

> attach(vigne)

Les objets suivants sont masqués from vigne (position 3):

    pepin.08, poids.08, pop, volcm3.08
```

Résumé statistique
Commande `summary`

Le résumé numérique s'adapte selon la nature des variables (univariée, multivariée, factorielle)

```
> summary(pepin.08)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00    1.40    1.80    1.86    2.40    3.20      27

> summary(pop)

CE CO TE
84 89 72

> summary(vigne)

 pop        pepin.08       poids.08      volcm3.08
 CE:84   Min.   :0.00   Min.   :0.39   Min.   : 3.4
 CO:89   1st Qu.:1.40   1st Qu.:0.82   1st Qu.: 7.1
 TE:72   Median :1.80   Median :1.06   Median : 8.9
         Mean   :1.86   Mean   :1.21   Mean   :10.5
         3rd Qu.:2.40   3rd Qu.:1.36   3rd Qu.:11.9
         Max.   :3.20   Max.   :3.75   Max.   :33.8
         NA's   :27     NA's   :28     NA's   :44
```

Résumé statistique

Package `Hmisc` commande `describe`

```
> library(Hmisc)
> describe(vigne)
vigne

 4  Variables      245  Observations
---------------------------------------------------------------
pop
      n missing  unique
    245       0       3

CE (84, 34%), CO (89, 36%), TE (72, 29%)
---------------------------------------------------------------
pepin.08
      n missing  unique    Mean     .05     .10     .25
    218      27      26   1.858     1.0     1.2     1.4
    .50     .75     .90     .95
    1.8     2.4     2.7     2.9

lowest : 0.0 0.5 0.8 1.0 1.1, highest: 2.8 2.9 3.0 3.1 3.2
---------------------------------------------------------------
poids.08
      n missing  unique    Mean     .05     .10     .25
    217      28     122   1.212   0.578   0.676   0.820
    .50     .75     .90     .95
  1.060   1.360   1.868   2.472
```

Tableau de fréquences

Dans le cas où la variable x est continue, la réalisation d'un tableau de fréquence nécessite un partitionnement préalable du domaine de définition en K classes de largeur

- constante
- ou variable

Fonction de répartition empirique

$$\hat{F} : \mathbb{R} \mapsto [0,1],\ x \mapsto \frac{1}{n} card\{i : x_i \leq x\}$$

- Le graphe de la fonction de répartition est une fonction en escalier appelé diagramme cumulatif

Graphe en tiges et feuilles
Commande `stem`

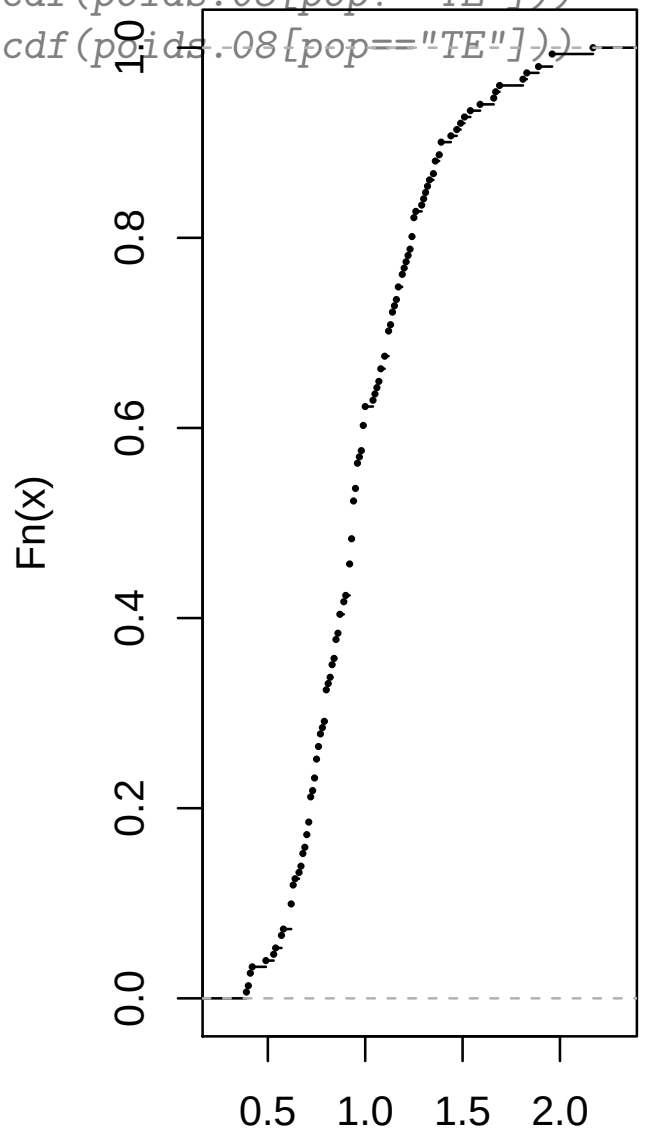
Un graphe en tige est feuille permet de visualiser le tableau des fréquences

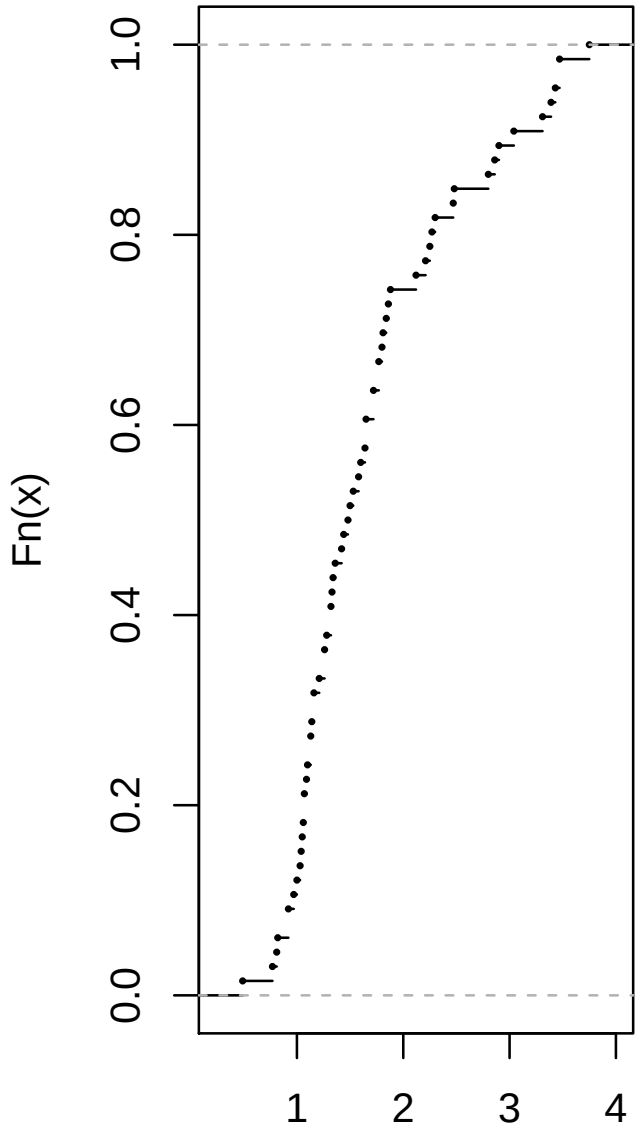
```
> stem(pepin.08)
  The decimal point is 1 digit(s) to the left of the |

   0 | 00000000
   2 |
   4 | 0
   6 |
   8 | 0
  10 | 00000000000
  12 | 00000000000000000
  14 | 0000000000000000000000000000000000000
  16 | 00000000000000000000
  18 | 00000000000000000000
  20 | 00000000000000000000
  22 | 00000000000000000
  24 | 00000000000000000000
  26 | 00000000000000000000
  28 | 00000000
  30 | 000
  32 | 0000000
```

Fonction de répartition empirique

`ecdf` crée un objet qui peut être tracé avec `plot`.

```
> par(mfrow=c(1,2))
> plot(ecdf(poids.08[pop!="TE"]))
> plot(ecdf(poids.08[pop=="TE"]))
```

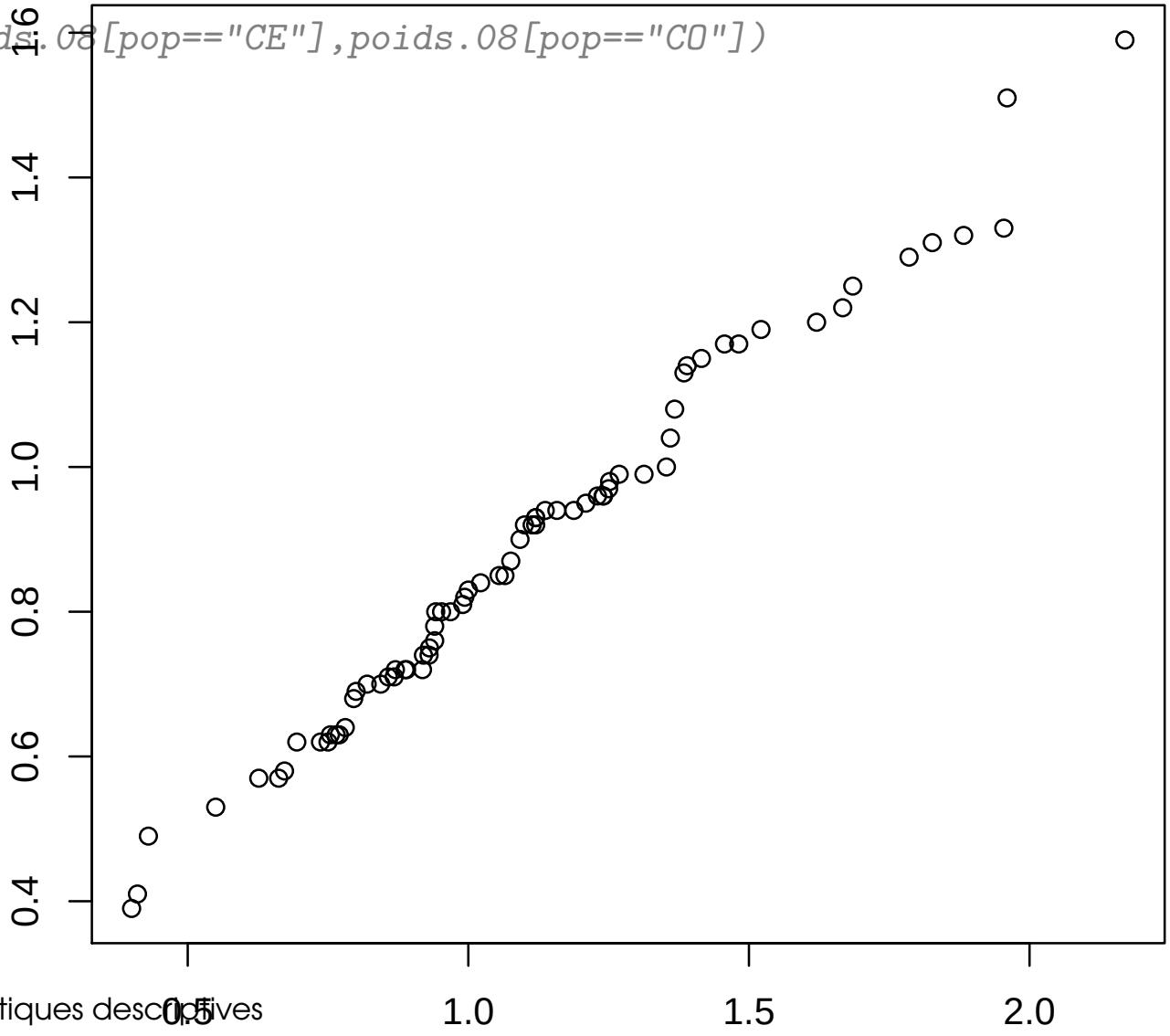




Comparaison de distribution

Pour comparer visuellemnt deux distributions, la manière la plus efficace est le graphe quantile/quantile (qui doivent correspondre si les distributions sont proches.)

```
> qqplot(poids.08[pop=="CE"],poids.08[pop=="CO"])
```



Histogramme I

- Estimateur de la fonction de densité

$$\hat{f}_n(x) = \sum_i h_i \mathbb{1}_{[a_i, a_{i+1}[}(x) \quad a_1 < \ldots < a_{k+1}$$

- Découpage en intervalles
- Calcul de la fréquence
- Aire du rectangle proportionnel à la fréquence

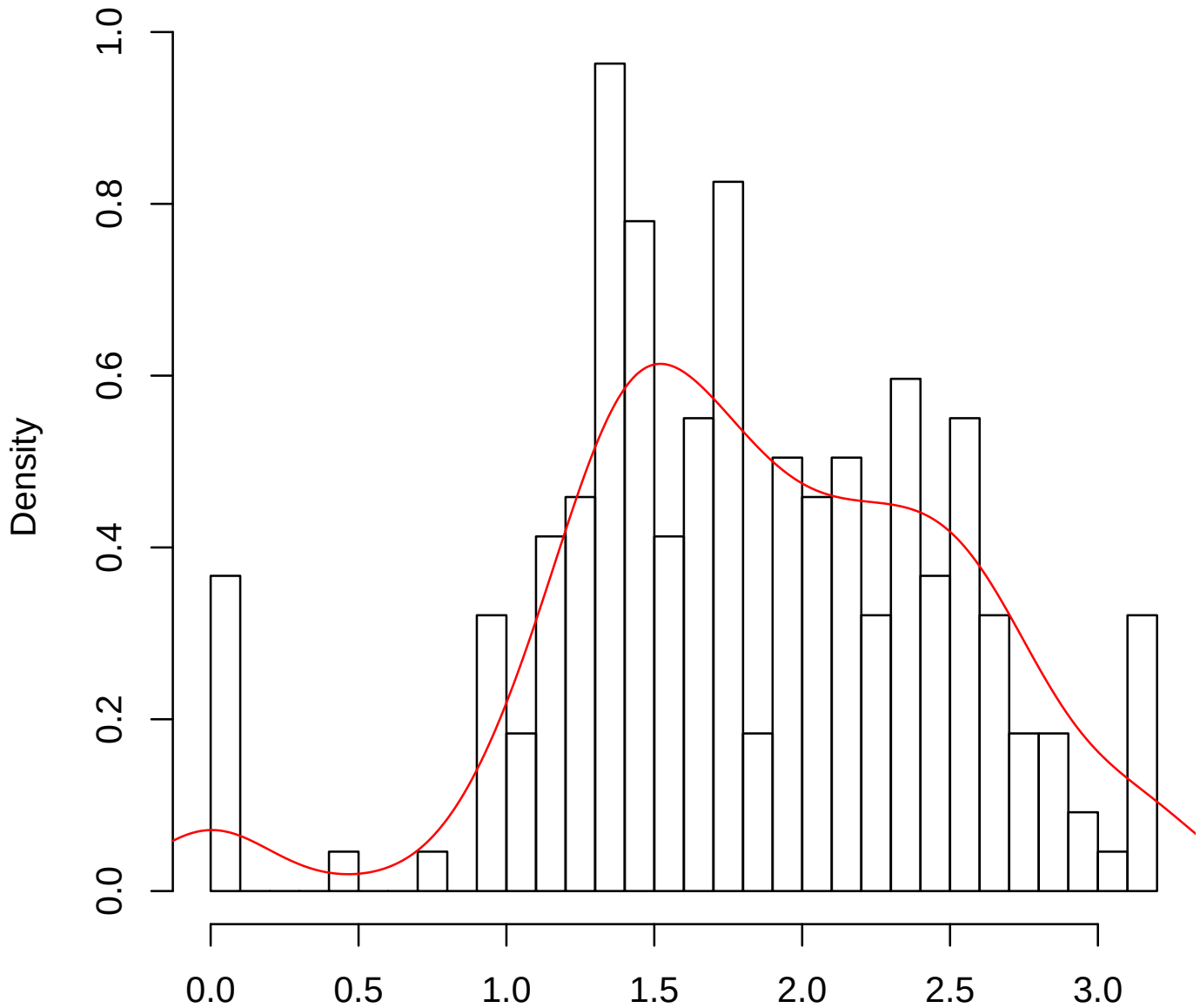
$$\sum_i h_i(a_{i+1} - a_i) = 1 \text{ et } h_i(a_{i+1} - a_i) = \hat{P}_F(X \in [a_i, a_{i+1}[)$$

Histogramme II

- Attention : hauteur proportionnelle à la fréquence si et seulement si les intervalles ont tous la même largeur
- Nombre d'intervalles :
  - Important
  - Réglage difficile
  - Règle empirique : règle de Sturges $1 + 10/3 * \log_{10}(n)$

Histogramme

```
> hist(pepin.08,nclass=25,prob=TRUE)
> lines(density(pepin.08[!is.na(pepin.08)]))
```



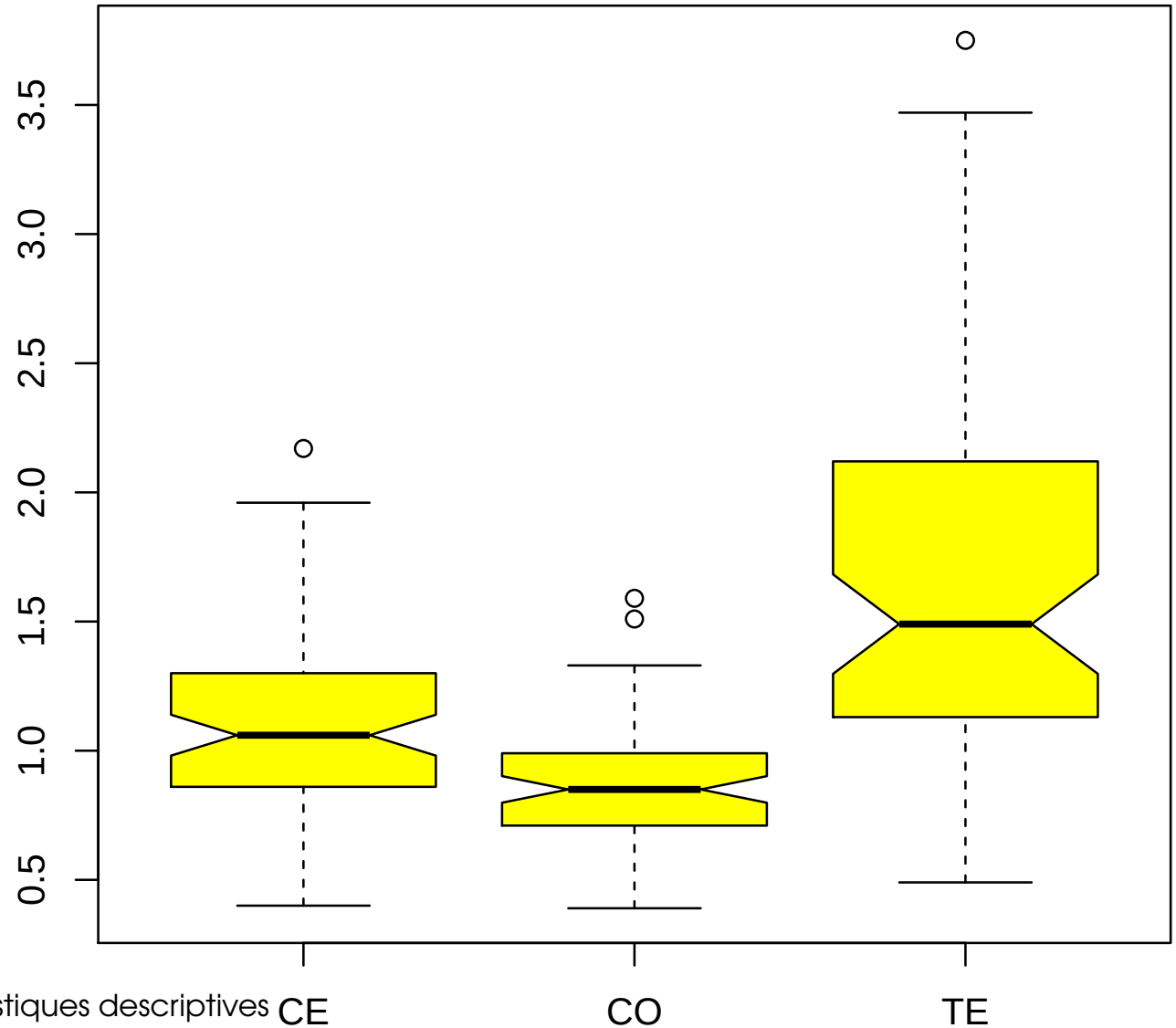
Boîte à moustaches ou boxplot

- Éléments atypiques (aberrants, *outliers*)
  - Notion arbitraire
  - Règle empirique assez souvent utilisée : valeurs situées à l'extérieur de $[q_1 - 1.5 \times Iqr, q_3 + 1.5 \times Iqr]$
- Définition : Graphique constitué
  - d'un rectangle délimité par les quartiles et partagé en deux par la médiane
  - d'une paire de moustaches : minimum et maximum de l'échantillon auquel on a ôté les éléments atypiques
  - des outliers eux-mêmes

Boîtes à moustaches

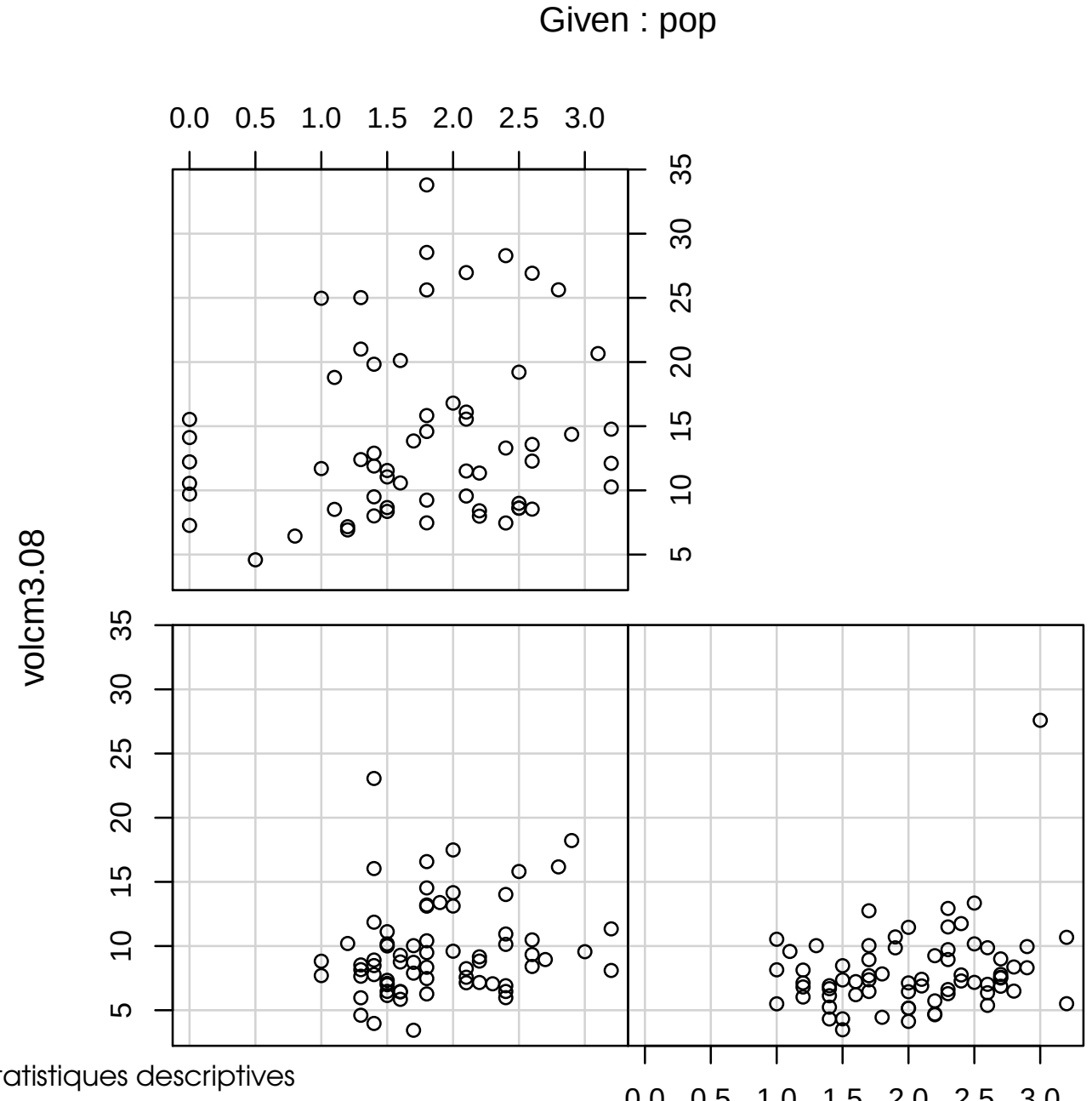
La boîte à moustache permet de visualiser les grands traits caractéristiques d'une distribution.

```
> boxplot(poids.08~pop,col="yellow",notch=T)
```



Graphe conditionné par une variable

```
> coplot(volcm3.08 ~ pepin.08 | pop, show.given=FALSE)
```

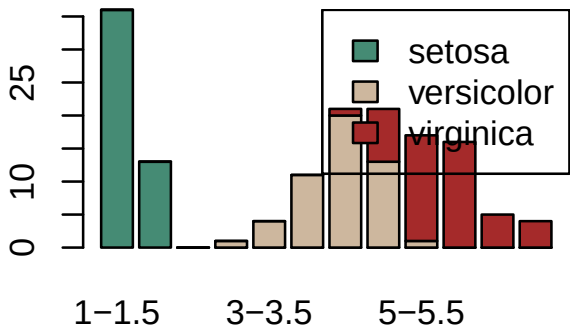


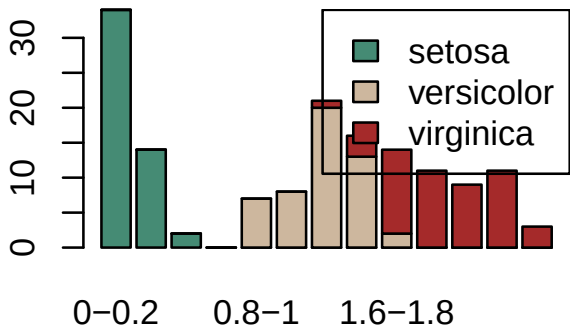
Conclusion

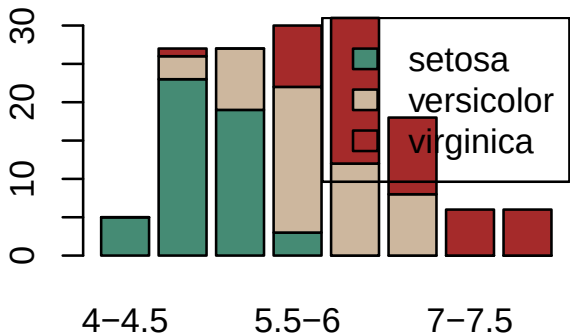
Mise en évidence de certaines caractéristiques :

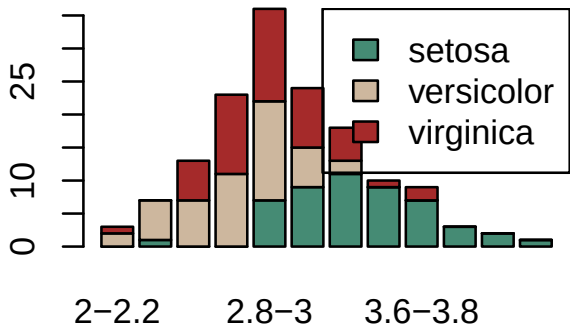
- Présence de données atypiques
- Absence de symétrie de la distribution
- Présence de populations hétérogènes
- ...

Histogrammes et variable qualitative









Statistiques associées à un vecteur aléatoire I

- Rappels
  - X réalisation d'un échantillon de taille n du vecteur aléatoire $\mathbf{X}$
  - $\boldsymbol{x}_i$ réalisation de taille 1 de $\mathbf{X}$
  - $\boldsymbol{x}^j$ réalisation d'un échantillon de taille n de X^j
- Moyenne empirique

$$\overline{\boldsymbol{x}} = (\overline{x}^1, \ldots, \overline{x}^p)' \quad \text{où} \quad \overline{x}^j = \frac{1}{n} \sum_{i=1}^{n} x_i^j$$

- Variance empirique

$$s_j^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i^j - \overline{x}^j)^2$$

Statistiques associées à un vecteur aléatoire II

- Covariance empirique

$$s_{jj'} = \frac{1}{n}\sum_{i=1}^{n}(x_i^j - \overline{x^j}).(x_i^j - \overline{x^{j'}})$$

- Coefficient de corrélation linéaire empirique

$$r_{jj'} = \frac{s_{jj'}}{s_j\, s_{j'}}$$

- Matrice de variance empirique

$$S = (s_{jj'}) = \frac{1}{n}(X - 1_n\overline{\boldsymbol{x}})'(X - 1_n\overline{\boldsymbol{x}}) = \frac{1}{n}\,Y'\,Y$$

où 1_n est la matrice de dimension $(n, 1)$ remplie de 1 et Y est la matrice centrée associée à X.

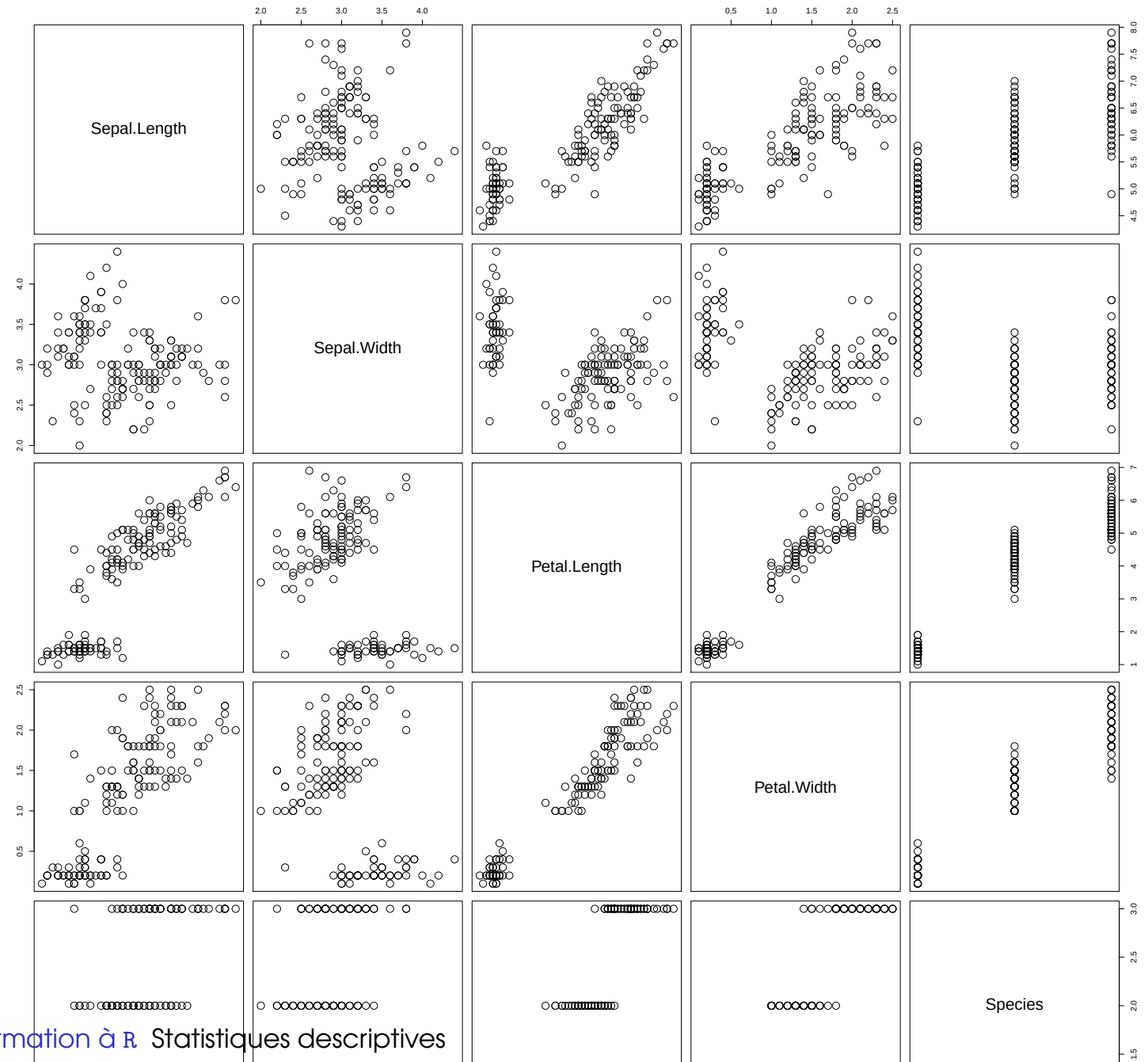
- Matrice de corrélation empirique

$$R = (r_{jj'}) = D_{1/s_j} S D_{1/s_j}$$

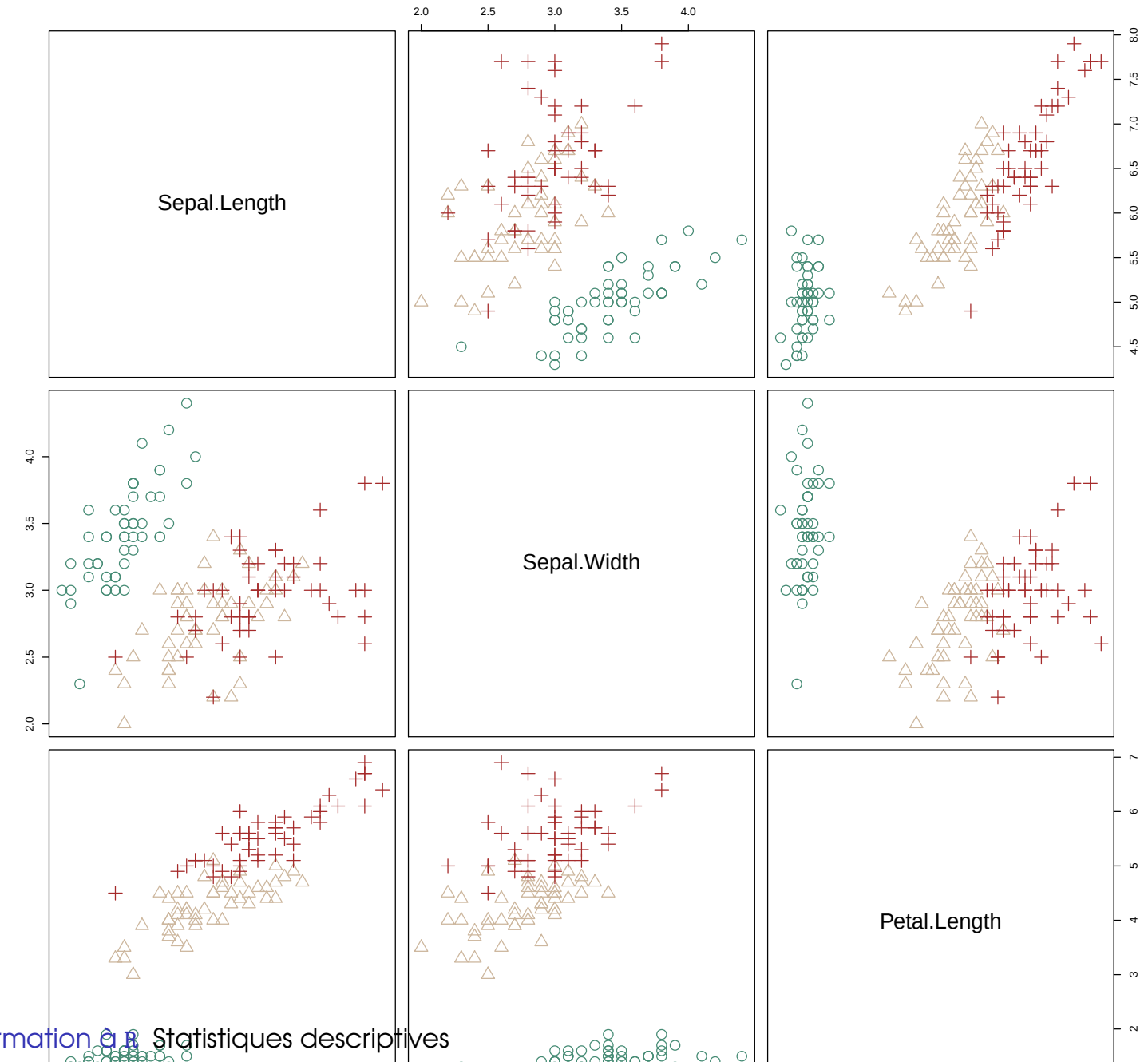
Graphique de dispersion

- Représentation de chaque individu i par le point du plan (x_i^1, x_i^2)
- Nuage de n points dans le plan
- Visualisation synthétique des données : permet de voir
  - les relations linéaires
  - les regroupements en classes homogènes

Les 5 variables des iris



Les 5 variables des iris en couleurs



Covariance et corrélation

- 2 variables : covariance et corrélation empirique
- > 2 variables : matrices de cov. et de corr. empiriques

Les iris

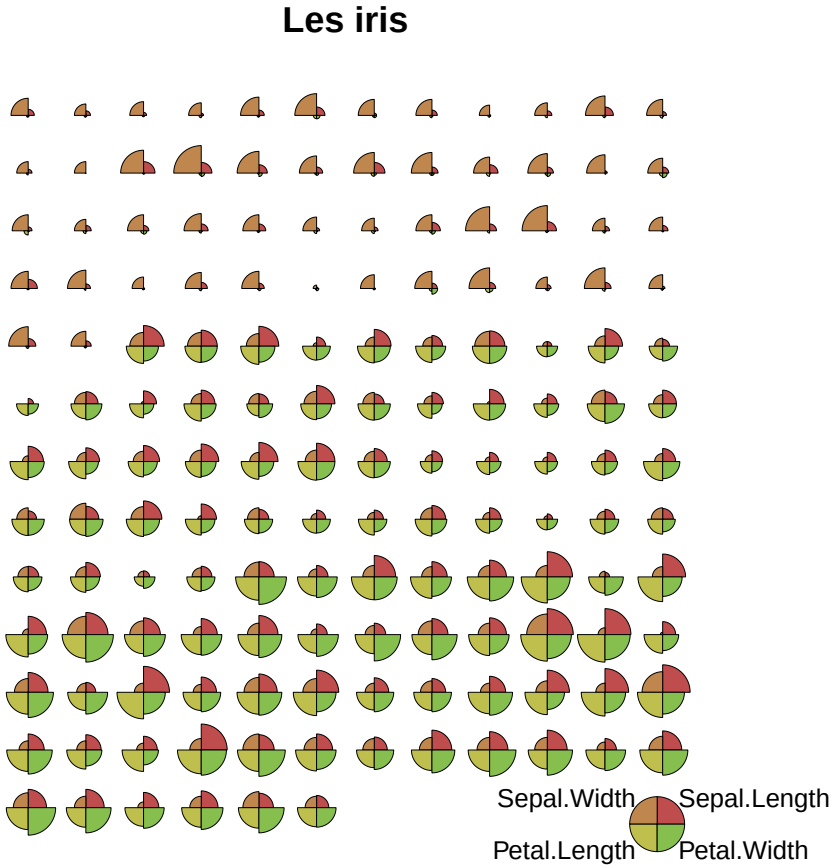
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.69	-0.04	1.27	0.52
Sepal.Width	-0.04	0.19	-0.33	-0.12
Petal.Length	1.27	-0.33	3.12	1.30
Petal.Width	0.52	-0.12	1.30	0.58

TABLE : Matrice de covariance

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.69	-0.04	1.27	0.52
Sepal.Width	-0.04	0.19	-0.33	-0.12
Petal.Length	1.27	-0.33	3.12	1.30
Petal.Width	0.52	-0.12	1.30	0.58

TABLE : Matrice de corrélation

Les diagrammes fleurs



Fléau de la dimension (curse of dimensionality) I

- Espace de grande dimension
- Calculs similaires à ceux du plan
- Mais difficile de généraliser
- Exemple 1 :
  - Dans $\mathbb{R}$
    - Pts uniformément répartis dans $[-1, +1]$
    - % de points situées à 1 distance ≤ 0.75 de l'origine : 75%
  - Dans $\mathbb{R}^{10}$
    - Pts uniformément répartis dans $[-1, +1]^{10}$
    - % de points situées à 1 distance ≤ 0.75 de l'origine : 5%
- Exemple 2 : on veut construire un histogramme en s'appuyant sur au moins une moyenne de 10 points par intervalle et 10 classes par variable
  - $\mathbb{R}$: 10 classes $n = 100$
  - $\mathbb{R}^2$: 100 classes $n = 1000$
  - $\mathbb{R}^{10}$: 10^{10} classes $n = 10^{11} = 100 milliards$

Fléau de la dimension (curse of dimensionality) II

- Si p assez grand, l'espace $\mathbb{R}^p$ est pratiquement vide et sauf si les données se situent au voisinage d'une variété de faible dimension, l'analyse des données n'apportera aucune information intéressante.
- Les points voisins d'un point donné sont tous très loin : difficultés dans l'emploi de méthodes du type k-plus proches voisins